

TESE DE DOUTORADO

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO

**“Model-Based Design of User Interfaces to Support
Situation Awareness in Maintenance”**

ALUNO: Allan César Moreira de Oliveira
ORIENTADOR: Profa. Dra. Regina Borges de Araújo

São Carlos
Novembro/2016

CAIXA POSTAL 676
FONE/FAX: (16) 3351-8233
13565-905 - SÃO CARLOS - SP
BRASIL

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

**MODEL-BASED DESIGN OF USER INTERFACES TO
SUPPORT SITUATION AWARENESS IN
MAINTENANCE**

ALLAN CESAR MOREIRA DE OLIVEIRA

ORIENTADORA: PROFA. DRA. REGINA BORGES DE ARAÚJO

São Carlos - SP
Novembro/2016

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

**MODEL-BASED DESIGN OF USER INTERFACES TO
SUPPORT SITUATION AWARENESS IN
MAINTENANCE**

ALLAN CESAR MOREIRA DE OLIVEIRA

Tese apresentada ao Programa de Pós-Graduação em
Ciência da Computação da Universidade Federal de
São Carlos, como parte dos requisitos para a
obtenção do título de Doutor em Ciência da
Computação, área de concentração: Processamento
de Imagens e Sinais
Orientadora: Profa. Dra. Regina Borges de Araújo

São Carlos - SP
Novembro/2016





UNIVERSIDADE FEDERAL DE SÃO CARLOS
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a defesa de Tese de Doutorado do candidato Állan César Moreira de Oliveira, realizada em 13/12/2016.


Prof.ª. Dr.ª. Regina Borges de Araujo
(UFSCar)



Prof.ª. Dr.ª. Vânia Paula de Almeida Neris
(UFSCar)


Prof. Dr. Orides Morandin Junior
(UFSCar)

Prof. Dr. Alexandre Cardoso
(UFU)

Prof.ª. Dr.ª. Danúbia Bueno Espíndola
(FURG)

Certifico que a sessão de defesa foi realizada com a participação à distância dos membros Alexandre Cardoso e Danúbia Bueno Espíndola, depois das arguições e deliberações realizadas, o participante à distância está de acordo com o conteúdo do parecer da comissão examinadora redigido no relatório de defesa do aluno Állan César Moreira de Oliveira.


Prof.ª. Dr.ª. Regina Borges de Araujo
Presidente da Comissão Examinadora
(UFSCar)

AGRADECIMENTOS

Primeiramente gostaria de agradecer minha orientadora, profa. Dra. Regina Borges de Araujo, por ser uma guia nesse período de amadurecimento como pesquisador.

Também gostaria de agradecer a todos os colegas do laboratório WINDIS (former LRVnet), Leonardo Botega, Rafaela Vilela, Claudia Berti, Fernando Duarte, Gislaine Michelloti, Antonio Dourado, Adriel Radichi, entre outros passageiros, pelo apoio no dia a dia e a paciência nos workshops.

Da mesma forma, sou grato aos colegas do C-MORE (Universidade de Toronto): meu coorientador na UofT, prof. Dr. Andrew Jardine, meus colegas de laboratório Dragan Banjevic, Neil Montgomery, Corey Kiassat, Dana Shaevitch, Janet Sung, Clayton Van Volkenburg, Maliheh Aramoon, Turuna Seecharan, Laurent Caudrelier, Soroush Sharifi, Erik Bjarnason and Stephan Trusevych.

Este trabalho não seria possível sem o suporte dos os colaboradores da Usina Hidrelétrica de Igarapava, que me ajudaram a aprender sobre o domínio da manutenção e possibilitaram desenvolver os testes do projeto.

Por fim, gostaria de agradecer aos amigos que me deram forças quando precisei, Daniel Macedo, Guilherme Sabino, Hudson Lazari, e à minha família que esteve sempre presente, principalmente minha esposa, Nahana Caetano, que me acompanhou durante toda a pesquisa no Canadá, meu pai, Júlio Caixeta, minha mãe, Ana Beatriz e minha irmã, Pâmela Oliveira.

RESUMO

Consciência Situacional (SAW) é um processo cognitivo definido como a percepção de elementos e eventos em um volume de tempo e espaço, o entendimento da situação atual e a projeção do seu estado futuro. SAW é um pré-requisito para a tomada de decisão em sistemas complexos e dinâmicos, e os problemas em SAW são uma fonte reconhecida de erros humanos e acidentes. O estudo de SAW não só é fundamental em muitos setores (como, por exemplo, aviação, forças armadas, óleo, gás e ferrovias), como vem sendo considerado, de forma crescente, em áreas como a manutenção, que é uma atividade crítica para todas as indústrias. Novas tecnologias para apoio a manutenção, como a E-Manutenção, vão prover melhor acesso a informações desejadas, porém a SAW de um trabalhador que lida com tecnologia está correlacionada a Interface do Usuário (IU) do sistema, e diversas IUs para manutenção apoiam a tomada de decisão considerando critérios procedurais e técnicos, mas não econômicos, legais, éticos e políticos. Portanto elas somente apoiam um desenvolvimento parcial da SAW de seu usuário, mas não a completa compreensão e projeção da situação. Estas interfaces ignoram requisitos de informação como: riscos e condições do ambiente; automações; ações e decisões de colegas de equipe; regras, regulamentos e políticas das empresas. Dessa forma, o design e desenvolvimento de IUs para aprimorar a SAW na manutenção são comprometidos pelas poucas soluções no estado da arte de design de IU para apoio a SAW, de processos de design baseado em modelos e de frameworks e arquiteturas de referência. Ciente desta lacuna, esta tese propõe uma solução para design e desenvolvimento de Interfaces do Usuário que apoiam o estabelecimento de SAW (SASUI) em trabalhos de manutenção. Três contribuições foram geradas: um framework conceitual de aspectos de SAW (FSA) que auxilia desenvolvedores a estruturar fontes de dados heterogêneas em um modelo de representação do conhecimento, para obter uma visão de SAW orientada a estado; uma arquitetura multiagente que instancia e controla IUs para aprimorar o apoio a SAW, usando uma combinação de agentes de SAW e IU que expressam a situação (e projeção) de entidades do mundo real; e, finalmente, uma metodologia para criar IUs para SAW baseada em modelos (MBSAW-UI), na qual designers modelam agentes que irão auxiliar usuários a adquirir a SAW necessária para seu processo de tomada de decisão. Estas contribuições seguem uma abordagem de Engenharia Cognitiva para guiar desenvolvedores de software no processo de design de IU e para permitir especialistas de domínio a modelar suas IUs, habilitando um paradigma de End-User Development (EUD) que facilita futuras atualizações ao sistema. Foi desenvolvido um estudo de caso de uma atividade de manutenção para avaliar as soluções propostas, usando duas interfaces: uma projetada usando a metodologia MBSAW-UI; e outra usando somente análise hierárquica de tarefas. Experimentos realizados mostram que as interfaces geradas com a metodologia proposta neste trabalho proporcionaram um aumento de 78% na SAW, o que levou a uma melhor eficácia (3,85x menos erros) e segurança (3,87x menos erros relacionados a comportamentos arriscados).

Palavras-chave: Consciência Situacional, Interface do Usuário, design centrado no usuário, Engenharia Cognitiva, design baseado em modelos, end-user development, sistema multiagente, agente de SAW, agente de Interface do Usuário, manutenção.

ABSTRACT

Situation Awareness (SAW) is a cognitive process defined as the perception of elements and events within a time frame, the understanding of their situation and the projection of their status. SAW is a prerequisite for decision-making in dynamic and complex systems and errors in SAW are an acknowledged source of human errors and accidents. Its study is pivotal in many industries, such as aviation, military, oil, gas and rails, and it is being increasingly considered for maintenance, since this activity is deemed critical for every industry. New technologies to support maintenance, such as E-Maintenance, will provide easier access to the desired information to technicians, but the way new technologies lead to improved SAW is influenced by how information is presented in the User Interface (UI), and many UIs for maintenance technicians support their decision-making regarding procedural and technical criteria, but not economical, legal, ethical and political. Therefore, they only allow a partial development of the user SAW, but not the complete comprehension and projection of a situation. These UIs ignore information requirements such as: risks and conditions of the environment; automations; actions and decisions of team members; rules, regulations and policies of enterprises. Therefore, the design and development of UIs to improve SAW in maintenance is compromised by the few solutions in the state of the art for SAW supportive UI design, for model-based design process and for frameworks and reference architectures. Cognizant to this gap, this thesis proposes a solution for the design and development of Situation Awareness support User Interfaces (SASUI) for maintenance work. For that three contributions are proposed: a conceptual framework of Situation Awareness Aspects (FSA) that assists developers in structuring heterogeneous sources of data into a knowledge representation model, to obtain a state oriented view of SAW; a multiagent architecture that instantiates and controls UIs to improve their support of SAW, by using a blend of SAW and UI agents to express the situation (and its projection) of real world entities in the UI; a methodology to create Model-based SAW User Interfaces (MBSAW-UI), in which designers model agents that will assist users in acquiring the SAW necessary for their decision-making process. These contributions follow a Cognitive Engineering approach to guide software developers in the UI design process and also empower domain experts to model their UIs, enabling an End-User Development (EUD) paradigm that facilitate future updates to the system. A study case of a maintenance activity was developed to evaluate these solutions, with two interfaces: a UI designed using MBSAW-UI; a UI designed using solely a Hierarchical Task Analysis. An experiment was performed and showed a 78% increase in SAW with the UI designed to support SAW, which lead to enhanced efficacy (3,85x less errors) and safety (3,87x less errors regarding unsafe behavior).

Key-words: Situation Awareness, User Interface, user-centered design, Cognitive Engineering, model-based design, end-user development, multiagent system, SAW agent, UI agent, maintenance.

LIST OF FIGURES

Figure 2.1: Model of SAW in dynamic decision-making, adapted from Endsley [Endley95].	26
Figure 4.1: Framework of Situation Awareness Aspects (FSA) for maintenance and its 7 awarenesses or aspects.	68
Figure 4.2: Personal Awareness in maintenance.	69
Figure 4.3: Procedure Awareness in maintenance.	70
Figure 4.4: Equipment Awareness in maintenance.	72
Figure 4.5: System Awareness in maintenance.	74
Figure 4.6: Environment Awareness in maintenance.	76
Figure 4.7: Team Awareness in maintenance.	78
Figure 4.8: Enterprise Awareness in maintenance.	80
Figure 4.9: Agent overview, with data being used to model decisions, which generate presentation (UI).	86
Figure 4.10: An example of agent with several decisions modeled from the same data.	87
Figure 4.11: Agent assemble/life cycle, going from the agent creation, through the definition (and implementation) of the agent internal data, to the modeling of decisions. .	88
Figure 4.12: Example of agents and the network formed by agents and situation aspects (awarenesses).	90
Figure 4.13: The view of the combined theories of SAW (Endsley 1995 model of SAW and DSA) in the autonomous SAW-driven interface agents.	92
Figure 4.14: A UI assisting the user to develop his/her SAW during three time frames. In the UI part, every concept developed in this thesis is explored: Endsley 1995 model of SAW, DSA theory, FSA and autonomous SAW-driven interface agents using a MAS architecture.	93
Figure 4.15 : an agent DFA with Transitions and using Messaging in an agent.	96
Figure 4.16: Architectural view of the reuse of agents in the system.	97
Figure 4.17: Logical architecture: scenario of agent instantiation.	99
Figure 4.18: Process architecture: scenario of agent communication between the local and virtual layers.	99
Figure 4.19: Development architecture based on 4 layers: system of system, virtual agents, local agents, UI.	100
Figure 4.20: The organization model of our MAS architecture, a modified hierarchical model, allows for: direct agent communication; agent to awareness communication;	

both master agent to optimize (layout manager) and decide (user) based on a global view resultant of the observation of all the agents.	101
Figure 4.21: Partition dynamism, due to the awareness division, creates a clear division in the layout to characterize types of data in fixed positions for users.	103
Figure 4.22: The two visualization modes of the UI. In (A) the Visible mode show important information, usually Local SAW (procedure) and new information and events (plus information personalized by the user). In (B) the Nonvisible mode shows all information available and relevant for the current situation.	103
Figure 4.23: Logical and process architecture: scenario of agent execution.	105
Figure 5.1: Overview of the methodology MBSAW-UI, for UI design and development to support SAW in maintenance work.	109
Figure 5.2: Model-Based SAW-oriented Requirements Engineering with stakeholders.	112
Figure 5.3: Example of GDTA for a corrective maintenance task.	114
Figure 5.4: Environment Awareness in the FSA of the maintenance domain.	116
Figure 5.5: All the types of Decision Components, their structure and the possible Messaging Mechanisms.	119
Figure 5.6: The creation of new UI components has three elements: (A) a set of data input, (B) a processing script and (C) a set of visual outputs using basic visual types.	122
Figure 5.7: Process of linking Data Components of the agent to data inputs of the UI.	122
Figure 5.8: Example of metadata in UI components describing information fields.	123
Figure 5.9: Run-time architecture implementation process.	123
Figure 5.10: Model-Based SAW-oriented UI Design with stakeholders.	125
Figure 5.11: GDTA for the task of diagnosing a broken generator.	126
Figure 5.12: Overview of the decision modeling processing.	128
Figure 5.13: A modeled Standard Decision Component, with Messaging schematics defined for each transition, and UIs defined for each state of each DFA.	130
Figure 5.14: Decision State Graph and the final UI, based on each DFA state.	130
Figure 5.15: Overview of the process of modeling a Standard Decision Component.	131
Figure 5.16: Complete information requirement of the decision “Which equipment should be inspected?”.	132
Figure 5.17: Level 2 SAW DFA with transitions for the decision “Which equipment should be inspected?”.	134
Figure 5.18: Level 3 SAW DFA with transitions for the decision “Which equipment should be inspected?”.	134

Figure 5.19: Defining the UI for the Decision Component, using only basic visual types and incremented Presentation components.	135
Figure 5.20: Defining static information in a UI.	136
Figure 5.21: Process of linking Data Components of the agent to the dynamic information of the UI.	136
Figure 6.1: Photos of the real environment of a hydropower plant generator. (A) a photo of a mockup of the generator bulb – side view; (B) a photo of the generator bulb from inside – front view; (C) a photo of the generator cover.	141
Figure 6.2: 3D models constructed to represent the power plant generator. (A) 3D model of the generator bulb – side view; (B) 3D model of the generator bulb – front view; (C) 3D model of the generator cover; (D) 3D model of the generator.	141
Figure 6.3: Hierarchical Task Analysis of a hydropower plant generator inspection.	144
Figure 6.4: Goal-Directed Task Analysis (GDTA) of the task of inspecting a hydropower plant generator.	145
Figure 6.5: Simulation control interface, to allow users to rest during their task.	147
Figure 6.6: Guide-UI for procedure 4 (inspecting the bulb).	149
Figure 6.7: Guide-UI and all the UIs for every procedure.	149
Figure 6.8: Schematic of the entities (agents) present in the study case/simulation. It shows agents of the type Environment (rectangles), Equipment (rounded rectangles), Procedure (circles) and Team members (triangles).	153
Figure 6.9: Inspection procedures definition for the study case, for Decision 1.2: How to inspect the equipment?, using the Procedure inspection SpecDC (Section 5.6.1).	155
Figure 6.10: UI generated for situation S1 of the simulation for the Decision 1.3 Is it necessary to fix the equipment (categorize the type of maintenance)?.....	157
Figure 6.11: UI generated for situation S2 of the simulation for the Decision 1.3 Is it necessary to fix the equipment (categorize the type of maintenance)?.....	158
Figure 6.12: UI generated for situation S3 of the simulation for the Decision 1.3 Is it necessary to fix the equipment (categorize the type of maintenance)?.....	158
Figure 6.13: (A) UI defined for Decision 2.2 What equipment / tools need to be used to prevent personal injury and environmental risks? and Decision 2.1 What are the risks to be prevented?; (B) UI defined for Decision 2.4 Which tool is required to execute the work?; Both UIs used the Personal Protective Equipment SpecDC (Section 6.4.1.4.1).	159
Figure 6.14: UI defined for Decision 2.5 Where should the collective protective equipment be deployed?, using the Collective Protective Equipment SpecDC (Section 6.4.1.4.2).	159

Figure 6.15: UI components with information for the Decision 1.1 Which equipment should be inspected?.....	161
Figure 6.16: Static information defined for the UI component of the Decision 1.1 Which equipment should be inspected?.....	161
Figure 6.17: UI transformations for each DFA state for the Decision 1.1 Which equipment should be inspected?, with (A) level 3 of SAW and (B) level 2 of SAW.....	162
Figure 6.18: DFAs for the Decision 2.6 What are the possible risks to be aware? regarding insalubrity risks.	164
Figure 6.19: UI for the Decision 2.6 What are the possible risks to be aware?. In (A) only UI components and in (B) static information.	165
Figure 6.20: Messaging mechanisms and UIs defined for each state of each DFA for the Decision 2.6 What are the possible risks to be aware?, considering the WBGT/noise as a risk.	165
Figure 6.21: View of the whole UI in Decision 2.6 What are the possible risks to be aware?, when situation “Leave when possible” is triggered by transition p4 (someone is interrupting the user).	166
Figure 6.22: Messaging schematic and UIs defined for each state of each DFA for the Decision 2.6 What are the possible risks to be aware?, considering the stairs as a risk.	167
Figure 6.23: Defining the Decision 3.1. What are my teammates doing that I need to be aware of?, using the Team inspection SpecDC.	168
Figure 6.24: DFA for the Decision 3.2 Is there an incident principle?, considering fire.	168
Figure 6.25: Messaging mechanisms and UI for each state of the DFA for the Decision 3.2 Is there an incident principle?.....	169
Figure 6.26: DFA, Messaging mechanisms and UIs for the Decision 3.2 Is there an incident principle?.....	170
Figure 6.27: UI to report inspected problems in the simulation.....	174
Figure 6.28: Comparison of number of errors committed per simulation using Guide-UI and SAW-UI	177
Figure 6.29: Number of errors committed per simulation by users with both UIs, categorized by Local SAW, Global SAW (Equipment, Team, Enterprise and Environment) and overall.	178
Figure 6.30: SART result for Guide-UI and SAW-UI (experts) and SAW-UI (non-experts).	180
Figure 6.31: Answer to five questions experts were asked related to usability and usefulness of SAW-UI, using the Likert scale.....	181

Figure 7.1: Users subjectively rated (Likert scale ranging from 2 to -2) if SAW-UI helped them understand the 5 situation aspects from FSA (Procedure, Equipment, Team, Enterprise and Environment).	186
Figure 7.2: Number of errors committed in each situation simulated by order of execution. Errors committed with SAW-UI are always lower than those committed with Guide-UI.....	188
Figure 7.3: Illustration of the error: “Did not coordinate with the team to execute another task when the equipment was being used”.	195
Figure 7.4: Guide-UI and SAW-UI comparison regarding the component of Understanding within the SART.....	197

LIST OF TABLES

Table 1.1: Examples of problems/decisions and their supporting criteria.	17
Table 2.1: Comparison of SAW measurement techniques.....	35
Table 3.1: Quality indicator for SAW support among many works in the literature and this thesis.....	59
Table 5.1: Example of a decision-activity table considering the activities of restoration and diagnostic and the decision from the GDTA of Figure 5.3.....	117
Table 5.2: Examples of agents of all situation aspects (awarenesses).	118
Table 5.3: Examples of Messaging schematics, with the mechanism, target and additional properties of each schematic.	137
Table 6.1: Three situations, each one with eight Abnormalities in Operative Conditions (AOC), were designed for the task of inspecting a hydropower plant generator.	146
Table 6.2: Relation between activities and decisions.	151
Table 6.3: Cost, risk and impact of every problem of the study case.....	156
Table 6.4: Cost and risk when problems occur in combination.	157
Table 6.5: Decision Components created to model the Decision 1.1 Which equipment should be inspected?, and the agents that have these components.	160
Table 6.6: Transition Messages schematics of the Decision 1.1 Which equipment should be inspected?.....	162
Table 6.7: Compilation of agents and Decision Components of the study case.	171
Table 6.8: Randomization of situations in the simulation and the UI for each user playthrough to create a randomized complete block design.....	175
Table 6.9: SART results compiling the average of answers from experts and non-experts users.....	179

LIST OF ACRONYMS

- AOCs** - *Abnormalities in Operative Conditions*
- AR** - *Augmented Reality*
- CRF** - *Cameleon Reference Framework*
- CTA** - *Cognitive Task Analysis*
- C2** - *Command and Control*
- CIM** - *Computation Independent Models*
- CMMS** - *Computerized Maintenance Management Systems*
- DC** - *Decision Component*
- DSA** - *Distributed Situation Awareness*
- EHCI** - *Engineering for Human Computer-Interaction*
- EUD** - *End-User Development*
- FSA** - *Framework for Situation Awareness Aspects*
- GDTA** - *Goal-Directed Task Analysis*
- HMD** - *Head-Mounted Displays*
- HTA** - *Hierarchical Task Analysis*
- HCI** - *Human Computer-Interaction*
- IMIS** - *Integrated Maintenance Information System*
- IoT** - *Internet of Things*
- MRO** - *Maintenance, Repair and Operations*
- MBD** - *model-based design*
- MBSAW-UI** - *Model-based SAW User Interfaces*
- MBDUI** - *Model-based UI Design*
- MVC** - *Model-View-Controller*
- MAS** - *multiagent system*
- OMA** - *Open Mobile Alliance*
- PIM** - *Platform Independent Models*
- PSM** - *Platform Specific Models*
- PromDC** - *Prompt Decision Component*
- SA** - *Situation Assessment*
- SAW** - *Situation Awareness*

SAOD - *Situation Awareness Oriented Design*

SART - *Situation Awareness Rating Technique*

SASS - *Situation-Awareness Support Systems*

SASUI - *Situation Awareness Support User Interface*

SpecDC - *Specialized Decision Component*

StanDC - *Standard Decision Component*

SME - *Subject Matter Experts*

TTL - *Temporal Tracing Language*

UIs - *User Interfaces*

SUMMARY

CHAPTER 1 - INTRODUCTION	13
1.1 Context	13
1.2 Motivation	15
1.3 Thesis Overview	18
1.3.1 Goals and objectives	18
1.3.2 Propositions	20
1.3.3 Scope	21
1.4 Reading Map	23
CHAPTER 2 - FUNDAMENTALS	24
2.1 Initial Considerations.....	24
2.2 Situation Awareness	25
2.2.1 SAW Models	25
2.2.2 What Situation Awareness is not.....	28
2.2.3 Team Situation Awareness	29
2.3 Design and development of a Situation Awareness Support UI (SASUI)	30
2.3.1 SAW demons.....	31
2.3.2 Situation Awareness-Oriented Design (SAOD).....	31
2.3.2.1 Goal-Directed Task Analysis (GDTA).....	32
2.3.2.2 Guidelines.....	32
2.3.3 SAW Evaluation.....	34
2.4 Industrial Maintenance	36
2.5 End-User Development and Model-Based Design.....	40
2.6 Final Considerations	41
CHAPTER 3 - STATE OF THE ART	43
3.1 Initial Considerations.....	43
3.2 Situation Awareness Support System and UI development	44
3.2.1 Models of Situation Assessment	44
3.2.2 SAW and computational agents	47
3.2.3 SAW guided development and guidelines.....	50

3.3 Model-based UI development for agents.....	52
3.4 User Interface for maintenance.....	55
3.5 Discussion and comparison of the state of the art of SASUI design.....	56
3.5.1 A Novel Architecture for Situation Awareness Systems [Baader09].....	60
3.5.2 Modelling situation awareness for Context-aware Decision Support [Feng09].....	61
3.5.3 Modeling Task Transitions to Help Designing for Better Situation Awareness [Villaren12]	62
3.5.4 User interface design for situation-aware decision support systems [Nwiabu12].....	63
3.6 Final Considerations	65

CHAPTER 4 - A MULTIAGENT ARCHITECTURE FOR USER INTERFACES

THAT SUPPORTS SAW	66
4.1 Initial Considerations.....	66
4.2 Framework for Situation Awareness Aspects	67
4.2.1 Personal Awareness.....	68
4.2.2 Procedure Awareness	69
4.2.3 Equipment Awareness	71
4.2.4 System Awareness.....	74
4.2.5 Environment Awareness.....	75
4.2.6 Team Awareness.....	77
4.2.7 Enterprise Awareness	79
4.3 Local and Global Situation Awareness	81
4.4 Autonomous SAW-driven interface agents.....	83
4.4.1 Agents Characterization	83
4.4.2 Agents Model	85
4.4.3 Conformity to the fundamentals of Situation Awareness.....	88
4.5 Model-based run-time environment architecture for SASUI	93
4.5.1 Agent acting to distribute cognition and for cooperative problem solving	94
4.5.2 Agent reuse	96
4.5.3 Dual run-time environment: a virtual and local layer.....	98
4.5.4 Agent independency and organizational model.....	100
4.5.5 Partitioned Dynamicity and Layers of information.....	102
4.5.6 Agent easy access to each other and componentized data structure.....	104
4.6 Final Considerations	105

CHAPTER 5 - MBSAW-UI: A MODEL-BASED DESIGN AND DEVELOPMENT

METHODOLOGY FOR USER INTERFACES THAT SUPPORTS SAW 107

5.1 Initial considerations 107

5.2 Overview of the Model-based SAW User Interfaces methodology 108

5.2.1 Involved stakeholders 108

5.2.2 Model-based Situation Awareness User Interface..... 109

5.2.3 MBSAW-UI in the software development process 111

5.3 Model-Based SAW-oriented Requirements Engineering..... 112

5.3.1 GDTA and Framework of SAW Aspects (FSA) 113

5.3.1.1 Creating a GDTA 113

5.3.1.2 Creating a FSA 115

5.3.1.3 Activities and the table of decision-activities..... 116

5.3.2 Agent identification and Situation Assessment (SA) Model..... 117

5.3.3 Presentation components design..... 120

5.4 Run-time architecture implementation 123

5.5 Model-Based SAW-oriented UI Design..... 125

5.5.1 Tasks and decisions 125

5.5.2 Decision modeling process 127

5.5.3 Standard Decision Component (StanDC)..... 129

5.5.3.1 GDTA requirements adjustment..... 131

5.5.3.2 High-level trigger 132

5.5.3.3 DFA 132

5.5.3.4 Levels 2 and 3 SAW dynamic equation 134

5.5.3.5 UI Components..... 135

5.5.3.6 Static information 135

5.5.3.7 Dynamic information..... 136

5.5.3.8 States UI transformation..... 137

5.5.3.9 Transitions Messaging schematics 137

5.5.3.10 Prompt Decision Component (PromDC)..... 137

5.6 Specialized Decision Component (SpecDC)..... 138

5.7 Final Considerations 138

CHAPTER 6 - STUDY CASE: INSPECTION OF A HYDRO POWER PLANT

GENERATOR 140

6.1 Initial considerations	140
6.2 Description of the simulated task: inspection of a hydro power plant generator	142
6.2.1 Task, domain and cognitive analysis	142
6.2.2 Simulation and situations	145
6.2.3 Efficacy and Situation Awareness analysis	147
6.3 Guide-UI development: focus on task analysis	148
6.4 SAW-UI development: a Situation Awareness Support UI (SASUI) developed using the MBSAW-UI methodology	150
6.4.1 Requirement Engineering	150
6.4.1.1 GDTA and FSA	150
6.4.1.2 Activities.....	150
6.4.1.3 Agent Identification.....	152
6.4.1.4 SA Model.....	154
6.4.1.5 Presentation components	154
6.4.2 UI Design.....	154
6.4.3 Iteration after SAW Measurement.....	170
6.4.4 Compilation of agents and Decision Components	171
6.5 Study case evaluation	172
6.5.1 Participants	172
6.5.2 Apparatus.....	173
6.5.3 Variables of the study and Experimental Design	174
6.5.4 Experimental Procedure	175
6.5.5 Results	177
6.6 Final considerations.....	181
CHAPTER 7 - CONCLUSIONS	182
7.1 Initial Considerations.....	182
7.2 Discussion of the thesis propositions	183
7.2.1 <i>PI</i> - design process	183
7.2.2 P2 - Situation Awareness support.....	184
7.2.3 P2.A - Global SAW	185
7.2.4 P2.B - SAW demons.....	186
7.2.4.1 Attentional narrowing.....	186
7.2.4.2 Requisite memory trap.....	187

7.2.4.3 Workload, fatigue and other stressors	187
7.2.4.4 Data overload.....	188
7.2.4.5 Misplaced salience.....	189
7.2.4.6 Complexity creep.....	189
7.2.4.7 Errant mental models.....	190
7.2.4.8 Out-of-the-loop syndrome	190
7.2.5 P2.C - SAOD guidelines.....	191
7.2.5.1 Organize information around goals.	191
7.2.5.2 Present Level 2 information directly	191
7.2.5.3 Provide assistance for Level 3 SAW projections	192
7.2.5.4 Use information filtering carefully	192
7.2.5.5 Manage rampant featurism through prioritization and flexibility	193
7.2.6 P2.D - SA model.....	194
7.2.7 P3 - applicability in the industry	195
7.2.8 P3.A - modeling capability	195
7.2.9 P3.B - adaptivity	196
7.2.10 P3.C - acceptability	196
7.3 Summary of contributions	197
7.4 Future works	199
REFERENCES	201
APPENDIX A - GDTA FOR MAINTENANCE.....	216
APPENDIX B - SPECIALIZED DECISION COMPONENTS.....	224
APPENDIX C - LIST OF ERRORS COMMITTED IN THE SIMULATION.....	232
APPENDIX D - QUESTIONNAIRE.....	235
APPENDIX E - PUBLICATIONS.....	240

Chapter 1

INTRODUCTION

1.1 Context

Situation Awareness (SAW) is the understanding of a situation by an individual for decision-making in dynamic and complex systems [Bossé07]. This cognitive process is considered one of the most important prerequisites for decision-making [Endsley89, Naderpour15].

SAW is one of the cornerstones of human-centered systems and is formally defined as “the perception of elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future” [Endsley95a]. It is currently applied in many industries, such as aviation, mining, oil/gas, rails and others, and lack of SAW (or errors in SAW) is a major factor in human errors (and accidents) in the industry [Endsley00, Bullemer13, Golightly13, Chinoy11, Lynas11, Sneddon13, Antonovsky14].

Studies in this area can deal with different issues, ranging from individual cognitive factors, such as information processing mechanism, mental model and pattern matching, to system and task factors, such as User Interfaces (UI), automation, complexity and workload.

New technologies to support work in industry, such as E-Maintenance, Augmented Reality and Head-Mounted Displays are expected to provide easier access to the desired information and thus improve SAW [Lipson98, Benbelkacem11, Espindola13, Enblom15]. Nevertheless, safety and reliability are not improved through mere exposure to data, and the way new technologies lead to improved SAW is influenced by how information is presented

[Endsley12]. A Cognitive Engineering approach is important to explore solutions for improved human system design that will lead to enhanced SAW [Endsley00a].

All of these problems are exacerbated in Maintenance, Repair and Operations (MRO), since, besides being an important activity present in every industry, this activity has an impressive impact on finances, personnel safety and product quality, having received intense attention within both industry and academia [Starr10, Tretten14].

Maintenance is characterized by industrial and technical processes/decisions, where tasks are executed in environments that demand high reliability and safety, involving a great amount of physical effort and immediate risks. These risks are measured by their effect on life, the environment and patrimony [Moreno-Trejo13, Robertson98].

To this day, many UIs for maintenance technicians (and other industrial workers) described in the literature have a restricted view of the work's process and environment, which is usually limited to the procedure itself instead of a more systemic approach [Friedrich02, Henderson10].

These UIs usually relegate cognitive demands, not supporting a holistic view of the work and ignoring information requirements such as: equipment risks; comprehension of the equipment state; automations; risks and conditions of the environment; ecologically sustainable development; correct procedure for waste disposal; actions and decisions of team members; rules, regulations and policies from enterprises (government, regulating agencies, the company).

Therefore, to increase SAW support, UIs should select data through established SAW-based guidelines. Ignoring these rules can potentially causes workers to under-perform, thus increasing the potential for the occurrence of human errors. It can also cause long-term rejection of new displays and technologies.

Current SAW-oriented methodologies provide guidelines for design [Endsley12, Stanton06] and computational Situation Assessment (SA) models [Matheus03, Jones11a, Hoogendoorn11b]. Many authors used them to design User Interfaces (UIs) to support SAW [Bolstad06, Romero-Gómez13, Botega15]. Nevertheless, to the best knowledge of this author, present academic research lacks model-based processes for Situation Awareness support User Interfaces (SASUI) design. To make matters worse, there are only a few frameworks or reference architectures for SASUI development.

A final challenge is enabling the End-User Development (EUD) paradigm for the industry, which can allow users (domain experts) to keep the system up-to-date by aligning it with changes in the tasks, organizational structure, environment or equipment [Ward08].

Accordingly, this thesis addresses the challenge of creating a model-based process to facilitate SASUIs design and development. In order to do so, it incorporates widely accepted SAW design guidelines, and integrates them with technological advancements and the EUD paradigm. By combining them, this work leads to improved decision-making by industry technicians, ultimately stimulating the adoption of SAW by stakeholders interested in system design for these maintenance technicians.

1.2 Motivation

This thesis is motivated by the needs of the industry to prevent and reduce accidents and increase process reliability, and by the necessity to improve methodologies for design of Situation Awareness Support UIs, since SAW is an important aspect of the decision-making in dynamic and complex systems. This section analyzes these motivations.

Lack of SASUI development techniques, frameworks, models and methodologies

Solutions to facilitate design/development of SASUI are the subject of only a few projects, despite its importance in UI design for many domains, [Stanton06, John08, Feng09, Chinoy11, Endsley12, Villaren12]. As stated by Naderpour et al., it is paradoxical that many human-system studies focus on technical elements, many times ignoring human factors such as SAW [Naderpour14]. Even the existing projects are targeted to Command and Control (C2) and none of them address specifically maintenance and industrial technical work, which have different requirements for system design.

Therefore, to guide this debate, first it is necessary to define the terms Situation Awareness Support UI (SASUI) and Situation-Awareness Support Systems (SASS).

SASS is a system that is aware of its users' Situational Awareness needs and requirements. SASUI, on the other hand, is basically a UI that supports users in acquiring and maintaining Situation Awareness through a variety of techniques. Of course, to some degree, any UI will support some level of SAW development. However, SASUI/SASS are UIs (and systems) specifically built to support it.

It is important to remind that these terms should not be confused with context-awareness (or context-sensitivity). A SASS is always aware of their user SAW development process, but it may or may not be context-aware.

In the ubiquitous computing community, the term situation-awareness is constantly used to define a higher level of context-awareness [Yau02]. In this thesis, however, this term will be used in the sense defined by the Cognitive Engineering community, as a cognitive process individuals have that is about being aware of what is happening around them and understanding the situation. Actually, one of the goals in this thesis is to make clear the separation of “cognitive” Situation Awareness [Endsley95] and computational situation-awareness [Yau02].

This confusion is seen in projects such as, Nwiabu et al. [Nwiabu12]. They used the term “Situation-aware user interface” and “situation-aware system” related to the “cognitive” SAW. The problem with their terminology is that it may influence readers to believe that the system or UI is the one that has SAW. However, users are the ones that have SAW, and the system/UI support them in obtaining SAW.

Lack of support for Global SAW in current UIs for maintenance work

Augmented Reality (AR), Head-Mounted Displays (HMD) and E-Maintenance UIs have a point in common: UIs are frequently created only for guidance of procedures [Henderson10, Yang15, Tretten11]. These UIs provide support for structured decisions based on procedural and technical criteria for decision-making (related to the execution of procedures). This type of support is classified as Local SAW support for the maintenance work domain.

In this domain, user attention should be on the physical world, especially on the manual operations he/she needs to execute. Therefore, the concept of a Local SAW being the procedure fits for this domain, since procedures are a “sequences of quantifiable tasks targeting a particular item in a specific location” [Henderson10]. In maintenance, they can be an “inspection, testing, servicing, alignment, installation, removal, assembly, repair, overhaul, or rebuilding of human-made systems” [Henderson10].

Nevertheless, most UIs were designed ignoring many problems that users face in their work outside the scope of the procedure and sometime even in the procedure, but considering different criteria for decision-making.

According to Vercellis [Vercellis09], there are six criteria that influence choices in a decision-making process: procedural, technical, economical, legal, ethical and political. Supporting Global SAW means supporting decision-making regarding this complementary set of criteria.

Common examples of problems/decisions and supporting criteria constantly ignored in the design of current UIs are listed in Table 1.1.

Table 1.1: Examples of problems/decisions and their supporting criteria.

Problem/ Decision	Supporting criteria (apart from procedural and technical)			
	Economical	Legal	Ethical	Political
Managing risks and safety during work	✓	✓	✓	
Following company and government regulations	✓	✓	✓	✓
Analyzing cost-effective solutions to problems	✓			
Working with a team	✓		✓	✓
Dealing with system automation	✓		✓	

Those examples listed in the last section represent ignored information requirements in UIs for maintenance technicians (equipment risks, automations, risks and conditions of the environment, actions and decisions of team members, regulations and policies from enterprises). They could, however, be provided by a UI supporting Global SAW. Additional analysis on Local and Global SAW in industry is presented in Section 4.2.

Demand for improvements of visualization solutions in maintenance

Obtaining and using information and knowledge in maintenance and operation is one of the challenges in the industry [Tretten14]. Although user-centered design and human-centered system concepts have been established for some time, UI for current systems in maintenance are still technology centered [Ward08, Jantunen11, Tretten14].

Furthermore, each company and its industries have their unique characteristics, which complemented by a stream of constant changes [Carvalho15], leads to experts constantly reworking Work Instructions to adapt and perfect them. In other words, the variability among industries prevents Work Instructions from being standardized.

Studies in maintenance systems acknowledge the importance of visualization techniques to improve workers' usage of information/knowledge [Bangemann06]. However,

this solution must also use the concept of End-User Development, to empower workers, usually domain experts, to adapt their computational resources, including User Interfaces, to the constant changes of the industry [Paterno13].

Additionally, UI design and development is known to be a time-expensive activity [Myers00], thus any improvement to speed up these processes is important.

Domain differential: maintenance and industrial work

Each domain has its unique characteristics and factors. Maintenance and industrial work are both characterized by: the low amount of concurrent goals and decisions for each task (despite there being a great amount of tasks, as explored in Appendix A); the dynamicity, unpredictability of the situation and constant evolvement of its processes (and changes in the equipment) [Carvalho15]; the presence of risks, measured by their effect on life, the environment and patrimony [Moreno-Trejo13]; the user (technician) focus on the physical world, where the work is executed.

These are some of the features to distinguish maintenance from Command and Control activities, a difference that also leads to different requirements for system design.

Despite its importance, UIs to support SAW are barely explored in the maintenance domain [Endsley00a, Nazir12]. In fact, as far as the authors know, SAW was never used as a theory to improve UIs in maintenance technical work. And while there are many projects using SAW in C2 [Romero-Gómez13, Botega15, Opto2213, Bolstad06], their findings are many times not applicable to maintenance. Therefore, due to the unquestionable relevance of SAW in maintenance and the industry [Endsley00a, Oedewald03, Bullemer13, Golightly13, Chinoy11, Sneddon13, Antonovsky14], it should be further explored for this domain.

1.3 Thesis Overview

1.3.1 Goals and objectives

The tools and solutions to guide developers and designers in creating a SASUI are almost inexistent and they do not support characteristics of the industry domain (high amount of tasks, situation unpredictability, risks, physical world restraints and effects). Furthermore, a

useful solution for this problem has to consider how to integrate itself with technological advancements such as E-Maintenance and Internet of Things.

Current systems for industrial maintenance support the user decision-making regarding procedural and technical criteria, but not economical, legal, ethical and political. Therefore, they allow only a partial development of the user SAW, but not the complete comprehension and projection of a situation. This deficiency reflect in an incomplete or erroneous SAW, leading to errors and accidents.

Therefore, the main goal of this thesis is to present and validate a solution for the design and development of Situation Awareness Support User Interfaces for maintenance work. The solution proposed in this thesis is a method to model User Interfaces to support Situation Awareness, and it can be divided in two parts.

The first part is a model-based design (MBD) process that has the following goals: instruct UI designers on how to model SASUIs; assist developers in structuring heterogeneous sources of data from the industry in a knowledge representation model; facilitate the design and development of SASUIs through a model-based strategy; define SAW in maintenance field work; allow domain experts to design new SASUIs (or update them), collaborating with UI designers, enabling an EUD paradigm.

This model-based SASUI design process implements the Situation Awareness Oriented Design (SAOD), proposed by Endsley and Jones [Endsley12], in a model-based process. Besides that, it uses two resources to facilitate SASUI development: a cognitive map, named FSA, which helps divide SAW laterally (among aspects of the situations) and produce UIs that support Global SAW; a new category of computational agents, the autonomous SAW-driven interface agents, allowing fast UI SAW-driven design. These agents merged UI and SAW agents to improve user-system cooperation for SAW acquisition and maintenance.

The second part is a run-time architecture that has two goals: adapt implementations strategies of SASS for C2 to the maintenance domain; provide a flexible solution for the industries requirements of constant system updates, increasing applicability of SASS in the industry for technicians; serve as a reference architecture for SASUI development in collaborative and dynamic industrial works.

This model-based run-time environment will instantiate User Interfaces using the results of the model-based process. It implements a multiagent system (MAS) architecture using the autonomous SAW-driven interface agents, which can access data from an E-Maintenance system and present it to guide users during tasks, while considering and supporting the user SAW.

This model-based run-time environment is an abstraction layer for E-Maintenance systems that allows an easier development of Views for users (maintenance worker) of these systems. It will facilitate the creation and maintenance of UIs that support users' SAW, for each single enterprise tasks.

To evaluate this thesis a study case was constructed with a complete UI for a maintenance work, equivalent to a real task in a hydropower plant. This UI was designed and implemented using the solutions of the thesis, and was evaluated and measured by user testing, and this testing focused on three main parameters: SAW score (measured by the Situation Awareness Rating Technique), efficacy and safety.

The MBD process was analyzed through a comparative analysis with the SAW quality indicators and its "demons" [Endsley12]. The quality indicators are indicative factors of potential SAW support by technological systems. The eight SAW "demons" are recognized influence factors that impair an accurate SAW by users [Endsley12].

Finally, one last goal of this thesis is to promote SAW in maintenance and industrial field work and UIs to assist developing SAW, due to its recognized importance by the industry and its potential of improving efficacy, safety (resilience) and unpredictability.

1.3.2 Propositions

Based on the literature review of Situation Awareness Support UI design and development and User Interfaces in industry (more specifically for the study case, which is industrial maintenance), this thesis puts forward a group of propositions to verify its validity regarding design process, final generated products, SAW support, context-awareness, modeling capability and applicability/acceptability in the industry.

The **General proposition of this study** is to provide support for the design and development of User Interfaces, for the industrial domain, that assist users in acquiring and maintaining SAW. The proposed methodology, architecture and conceptual framework facilitate the development and design of Situation Awareness Support UIs, assisting developers to create systems that integrate with the process of maintenance technical work, with UIs that support SAW, both Local and Global, and safety, and deals with the unpredictability in the industrial scenario by enabling the extension/modification by end-users (experts) of UIs for tasks.

Such statement is investigated through the following specific propositions:

P1 (design process): the solution proposed concretize a valid vision for design and development of Situation Awareness Support UIs for maintenance work, by attending the quality indicators for SAW support, and the final product (UI) could be directly translated to a real case scenario to improve efficacy and safety of the technician (specially compared to current systems based on manuals).

P2 (Situation Awareness support): the model-based process generates UIs that support users in acquiring and maintaining Situation Awareness.

P2 is divided in the following subpropositions:

P2.A (*Global SAW*): the final product (UI) assist users in acquiring and maintaining Global SAW while not impairing Local SAW.

P2.B (*SAW demons*): the final product (UI) assists users in avoiding frequent problems with the SAW “demons”.

P2.C (*SAOD guidelines*): the design process facilitates following the guidelines of SAW oriented development from the literature.

P2.D (*SA model*): the Situation Assessment model, based on a multiagent architecture with autonomous SAW-driven interface agents, improve UIs support to SAW in the industry domain.

P3 (applicability in the industry): the model-based process develops products prepared for the industry domain.

P3 is divided in the following subpropositions:

P3.A (modeling capability): The End-User Development approach allow the creation, by users, of new UIs for tasks, beside modifying existing ones, which supplies users with their own manutenability strategy for the system, leading to a reduced impact in the constant changes of the industry.

P3.B (adaptivity): the final product (UI) has adaptivity to a wide range of situations.

P3.C (acceptability): the final product (UI) is well-accepted by industry workers.

1.3.3 Scope

In order to elucidate the framework for this thesis, this chapter aims at explaining its scope. Considering areas of study, this thesis is both within the area of Engineering for Human Computer-Interaction (EHCI) and in that of the Cognitive Science. This integration between HCI, Cognitive Science, System engineering and Human Factor areas belongs to the Cognitive Engineering field [Gersh05, Bossé07].

Although since the emergence of this term, many different definitions were proposed, there was always one commonality: the design of complex and interactive system must take a holistic and ecological stance, considering people, socio-technical interactions, artifacts, human goal, human cognition, automations and the environment.

From the point of view of the Cognitive Science and applied Human Factors, the work is grounded in the theory of Situation Awareness, a theory derived from the Information Processing Theory.

In terms of Systems Engineering (Software Engineering), this work uses the concept of Model-Based Design (also known as Model Driven Engineering), which makes the connection from high-level processes to implementation frameworks.

Finally, the area of Human Computer-Interaction (HCI), which studies interfaces between humans and computers, is the area this thesis is producing direct results, through techniques from the others two areas.

Therefore contributions from this thesis can be classified directly as Cognitive Engineering's contributions, and indirectly as EHCI and Cognitive Science contributions.

This thesis aims at three distinct targets: software developers (programmers and UI designers) and two types of end-user, Subject Matter Experts (SME, also known as domain experts) and maintenance and industrial technicians.

The first target are software developers and UI designers (also system designers). Both can reuse the architecture and models that will be presented, to facilitate the creation of UIs that support Situation Awareness. Programmers can implement the model-based architecture to create Views (MVC) for industrial systems (E-Maintenance). UI designers can work through the defined MBD process to specify new UIs for these systems.

The second target are end-users of the type Subject Matter Experts. They can utilize the solutions presented in this thesis to increase the content of digitalized Work Instructions, as long as developers and UI designers have provided the architectural software solution elaborated in the thesis. This means that for each enterprise, domain experts within the company can convert today hundreds to thousands of Work Instructions to a digital format using the model-based SASUI design process proposed in this thesis. Furthermore, by using this process, generated UIs for these Work Instructions will support users Situation Awareness. These users will be referred in the thesis as SME users.

The last target are end-users of the type maintenance and industrial workers. These workers will benefit from the UIs by having up-to-date SAW, which leads to improved efficacy and safety. These users are referred by many names in literature, such as technicians,

field workers and industrial workers. In this thesis they will be referred interchangeably as maintenance technicians, technicians, workers or simply as users.

1.4 Reading Map

Chapter 2 revises the Fundamentals of SAW, maintenance and SAW and MBD. The SAW model adopted is thoroughly explained, as is the current major process to design systems considering SAW, the SAOD [Endsley12].

The state of the art of maintenance UIs, SAW-oriented development and MBD for UI agents are covered in Chapter 3. This chapter also presents a debate of similar projects compared to this thesis considering SAW quality indicators.

Chapter 4 presents an overview of the autonomous SAW-driven interface agents and the Model-based run-time environment architecture based on these agents. This chapter also presents the Framework for Situation Awareness Aspects (FSA).

The methodology to design and develop SASUIs is explained at Chapter 5. Each step of the methodology is addressed with examples for developers and designers of UI.

Chapter 6 introduces a study case developed using the methodology and architecture proposed and Chapter 7 the evaluations and results of this thesis.

Finally, Chapter 8 presents conclusions and last remarks.

Chapter 2

FUNDAMENTALS

2.1 Initial Considerations

Situation Awareness (SAW) is formally defined as “the perception of elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future” [Endsley95a].

SAW is essential for many different work domains, including maintenance and industrial work, and can improve decision support systems in light of the new technologies, improve human information processing and the cognitive fit of humans and system [Bossé07]. According to Endsley and Jones, SAW [Endsley12] increases the decision-making capacity, inducing a reduction in errors and also affects significantly user safety at work, efficacy and effectivity.

Thus, SAW is the first concept introduced in this chapter. The model of SAW adopted in this thesis is briefly explained, followed by some observations of what is **not** SAW and how SAW functions in team works. Then, the fundamentals of SASUI design are presented, including a review of techniques to assist SAW design and evaluations methods.

Furthermore the unique characteristics of SAW in industry are debated. These are important because this thesis will correlate these characteristics with standard SAW design techniques to produce its model-based SASUI design methodology.

Finally, an explanation of End-User Development and Model-Based Design is offered, since the methodology proposed in this thesis uses both concepts.

2.2 Situation Awareness

Human-centered systems for industry are highly encouraged to use SAW-oriented guidelines to maintain users aware of team work, rules of engagement, automated processes, nuances and dangers of the work environment and equipment condition.

Before presenting the fundamentals of Situation Awareness Support UI (SASUI) development, the following sections introduce an in-depth explanation of SAW models and features.

2.2.1 SAW Models

The Endsley 1995 model of SAW is highly dominant in the literature and industry [Golightly10, Wickens08, Bossé07, Endsley15] and will be the one explored in this thesis, although some concepts from Distributed Situation Awareness Model [Stanton06] are also used.

Figure 2.1 is used to explain the concept of SAW according to Endsley's model, which shows that people are active participants in creating their own SAW, by directing attention, communicating, using tools and changing strategies to process information.

The SAW module represents the state of mind or knowledge of the Situation of users. This module is the product of the process represented in the figure: at each iteration of the user observation and cognitive processing of the real world, SAW will be updated.

In this dynamic model, people iterate gathering and interpreting information to form a model of the situation that will lead to search for more information, until they are satisfied enough with their understanding of the situation to make a decision and act. The name of this described process is Situation Assessment (SA). The result of SA (a process) is SAW (a state or product).

Going deeper inside the SAW module, there are distinct three levels. The differentiation between these levels is important because they point to different cognitive operations and can be addressed by different training and system design strategies [Wickens08].

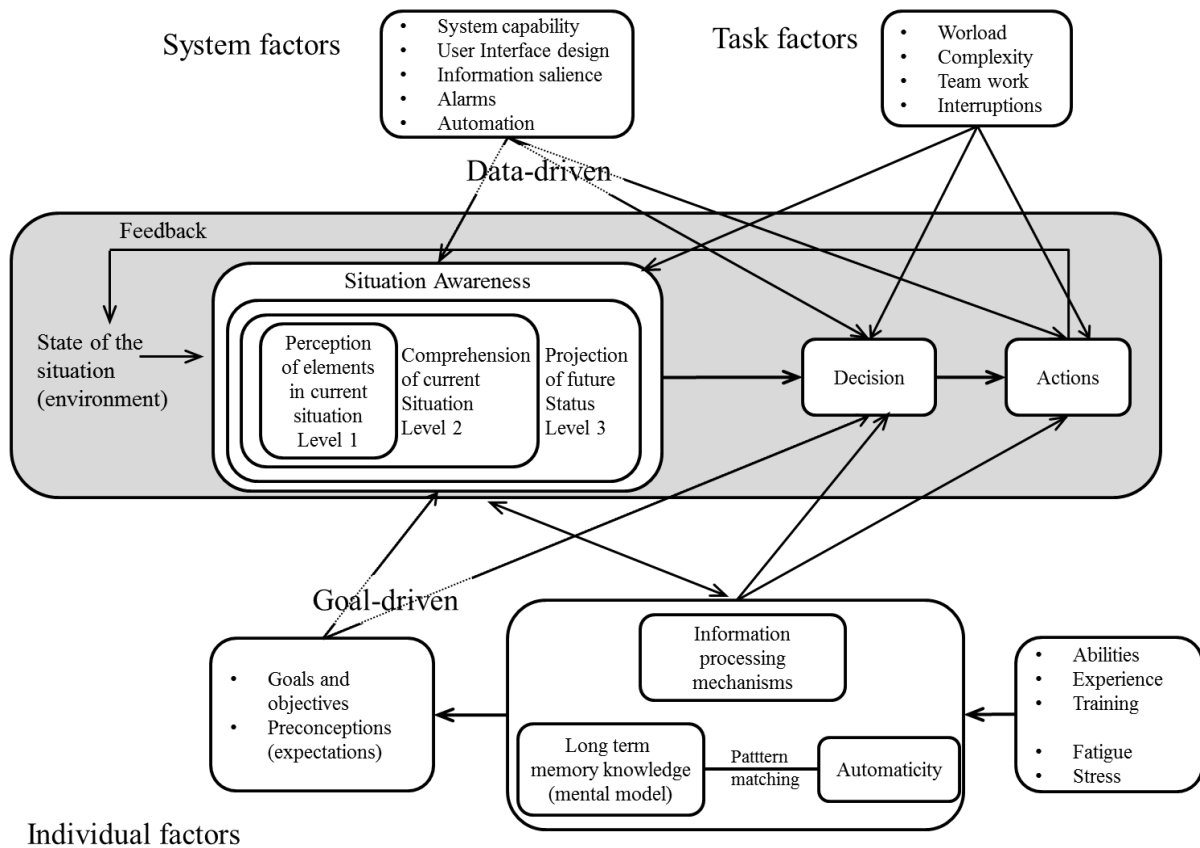


Figure 2.1: Model of SAW in dynamic decision-making, adapted from Endsley [Endley95].

The first level is **perception**, which is noticing the “environment” elements and their values, for instance an airplane altitude, an equipment temperature, or even the presence (or absence) of an alarm.

The second level is **comprehension and understanding**, and it is when a person aggregate several perception cues into a higher level situation. For instance, an airplane altitude by itself, means nothing, if the user cannot understand “what is altitude”, how does the current altitude constitute my situation (am I higher then I should be? Am I lower?).

The third level is **projection**, which is predicting the consequences of the current situation for a near and far future. It is usually the result from aggregating different perception and situations and their future impact to the system. For instance, “if the airplane keeps this altitude, what will it happen?”, “if it is too low, how much time can it keep going?”; “is there an obstacle ahead, and how far is it?”; “am I on a collision route?”.

Higher levels depend on the lower levels for accuracy, but the model is not necessarily linear [Endsley15a]. The three levels are actually characterized as ascending levels. A simple data-driven strategy (linear stages on level 1, 2 and 3) is not fitting for complex and dynamic systems, and that is why people use expertise and goal-driven strategies to fill in gaps.

A person with understanding of the current (level 2) and future (level 3) situation may not know all details (level 1) of the situation, but they can still make sound decisions. Since they may have to assume perceptual information (level 1), the knowledge of the current situation (level 2) is a drive for searching more details (level 1), to confirm their assumptions.

It can be noticed (actually most of the times), people are confident they understand and know “something”, but they don’t actually know every single part of it. A simple example could be the rain. Anyone can look at the sky and say if it is raining, and most people can even foresee a rain. However, not many understand and know every single variable that constitutes rain, and even more, their exact values at any time.

Continuing with Figure 2.1, Situation Awareness (its product) feeds the whole decision-making process. When SAW is perfect, it induces and potentiates (but does not guarantee) users to make the best possible decision for the situation. Without SAW users can still make the right decision, but it is a product of guess. With SAW, users can still make the wrong decision by lack of experience, lack of goodwill and other human and individual factors (impulsiveness, stress, etc) [Endsley00a].

Decisions lead to actions performed, which will result in changes in the world and the situation, and restart the process.

This SAW acquisition process is influenced by many internal and external factors. As a model inspired by the information processing theory, SAW processes consider the short-term (working) memory. Short-term memory is used to store temporary information about the elements of the situation. It is acknowledge to be limited to seven (plus or minus 2) chunks (information encoded to a more abstract level) of information [Miller56].

While this process begins with the working memory, information is compared against a knowledge model that is in the long-term memory knowledge to arrive at understanding or projection levels (2 and 3). This knowledge is called mental model, which is a mapping of the user knowledge of “how something works”. This mental model in return, lead to user expectations, that guide users to search for new information or to increase the certainty of information, influencing the level 1, 2 and 3 processes of SAW. Mental models are acquired through a lot of experience and training.

Novices do not have a complete mental model of the situation and have to rely heavily on working memory to store and process all information. Experts can use pattern matching to prototypical schemas to circumvent these limitations, quickly gaining comprehension of the situation even when it is complex or big (in data volume), in an almost automatic way (automaticity).

Another influencer of the quest for information is the current goal. Different goals lead to a direct search in the environment for appropriate information. Goal-direct processing directs user attention in the environment.

External factors also change and influence the Situation Awareness process. Different systems and their User Interface lead to different SAW, even if the situation is the same, because people is influenced by the way information is presented and which information is presented [Endsley12]. Others examples of influencers are: complexity of the task; automation; teams and environmental conditions.

This model also makes clear the importance of alternating data-driven and goal-driven processing of information. Current goals and expectations lead to a goal-driven search and processing of information, with users working in a top-down scheme, looking for information related to their objective. On the other hand, data-driven starts from the data to remodel the current goal, such as an alarm or information salience and highlight, catching the user attention and changing the task, in a bottom-up strategy. Attention will be directed by the current strategy, and alternating them, and providing support for this exchange, is vital for supporting users' SAW.

2.2.2 What Situation Awareness is not

It is also interesting to characterize what is not SAW, so that “good performance” is not classified as good SAW and so that this does not become a folk or vacuum concept (a concept that can engulf anything). There are four main characteristics that get confused with SAW, so they will be listed and explained as what is **not** SAW [Wickens08]:

- SAW should not be confused with many concepts from psychology, such as automaticity, working memory, preattentive processing, perception, confidence, goals, plans, scripts, mental models, schemas, pattern matching, expectations, information processing mechanisms, abilities and others.
- SAW is not action or performance: SAW is knowing the state of the situation (usually referred as “environment” in the literature, but this simplification term may hurt the concept, as explored in Section 2.4) or simply knowing what is going on, but is not a form of action. It is like the difference from diagnostic (SAW) to treatment (action). An operator with good SAW of a failing system understands the situation, but may not necessarily know how to counteract it, by not knowing the exact procedure or even not having the necessary motor skills to execute it (or the appropriate tools);

- SAW is not the same as long-term memory knowledge (or scripts, schemas and mental models): SAW is dynamic and is happening over the span of seconds, minutes, and at most hours. It is being conscious of the situation now. It should not be confused with knowledge, which is a mental model of the past user experience, mapping possible situations and outcomes. Static knowledge (declarative or procedural) is not SAW.
- SAW is not necessarily on working memory: SAW of experts works by instantiating sensorial inputs in the working memories as pointers to long-term memories. In this way, it is possible to surpass working memory limitations, especially when processing information to arrive at comprehension and projection of the situation [Endsley00a]. Experiments proved that even after 5-6 minutes, expert users were still retaining SAW information.

Nevertheless, as will be presented in Section 4.2, routine procedures and knowledge should be supported as soon as they become part of the current situation, when the topic under debate is about UIs to support SAW.

2.2.3 Team Situation Awareness

Teams are a group of people that work together towards a common goal. The concept of a team establishes a shared goal, interdependence of actions and labor division (responsibilities) [Endsley00a].

Team Situation Awareness is defined as “the degree to which every team member possesses the situation awareness required for his or her responsibilities” [Endsley89].

In a teamwork scenario, whenever a teammate needs information from another, a link between them is established, and this link is the Team SAW. Information needed by two or more people is referred as shared SAW. Information only needed by the individual is classified as individual SAW.

A high-level of Team SAW is important for team performance. It does not matter if a team member has all information he/she needs, if this information, when required, is not shared among team members.

Failures regarding Team SAW can even occur on the same three levels of SAW [Endsley00a]. In level 1, information is not successfully transmitted. In level 2, information/situation is not comprehended equally. In level 3, implications of future state are differently projected among team members.

Most errors occur when a team member assumes that the other teammate is aware of an information, when in fact he/she is not. For instance, in 1989, a Boeing aircraft crashed in

the United Kingdom, because one engine was on fire, but pilots turned off the wrong one. Later it was discovered that crew and passengers knew the right engine, but did not pass this information to the pilots, assuming that the pilots already knew it [United Kingdom Air Accidents Investigation Branch90].

As a lesson, if the person that needs the information does not have it, it does not matter that someone else on the team has it. Information needs to arrive at the person that needs it.

This concept is further explored by Stanton et al. and Stewart et al. with a description of SAW that is system-oriented, instead of individual-oriented [Stanton06, Stewart08]. Their theory of Distributed Situation Awareness (DSA) views a system where each agent (human or artefact) has awareness of a subset of knowledge objects (entities). A network of knowledge objects is naturally formed in a system (propositional network) and for each task a set of objects have to be invoked. Agents are then responsible for passing this knowledge for those interested. Therefore the network links (the passing of knowledge) are more important than the nodes themselves.

2.3 Design and development of a Situation Awareness Support UI (SASUI)

SAW is a part of the Cognitive Engineering (or Cognitive System Engineering), a multidisciplinary field that studies the relation of humans and systems. Other concepts researched and applied actively, regarding Cognitive Engineering, are: attention; decision-making; trust/complacency in automation; workload.

A Cognitive Engineering method for development of a support system has usually the following steps [Bossé07]:

- A cognitive analysis executed mainly through interviews with SMEs, followed by an identification of decision-making requirements;
- Choosing adequate cognitive models to analyze the decision-making requirements;
- Identify solutions to improve decision-making, which can come from enhanced technologies, systems, processes, UIs, organizational models, etc;
- Evaluation of the solutions to validate them from a human-performance and operational perspective.

The methodology for SAW design that grounded this thesis has similar steps and it begins by analyzing the influence of the SAW "demons" (i.e., factors that influence SAW in a negative way), presented in the next section.

2.3.1 SAW demons

Before presenting the most influential method for SASUI design and development, it is important to understand the typical SAW problems for users. The eight most studied factors that influence SAW in a negative way, known as the "SAW demons", are listed below [Endsley12]:

1. Attentional narrowing: when users stop switching attention between several information sources and focus on just one part of the whole.
2. Requisite memory trap: relying heavily on short-term memory.
3. Workload, fatigue and other stressors: stressors can strain SAW acquisition and maintenance.
4. Data overload: input of more information than what can be processed by the brain.
5. Misplaced salience: attention requested by an element, exacerbated by color, movement or noise, on the wrong place.
6. Complexity creep: an exaggerated number of features cause difficulty to form a mental model of the system.
7. Errant mental models: incorrect interpretation of the situation, ignoring data that does not match the user's expectations.
8. Out-of-the-loop syndrome: excess of automation make users unable to form a clear mental model of the system.

These factors are the initial step for achieving a process for SASUI development. Since they jeopardize SAW acquisition, if such a process does not consider these demons, it is prone to fail. Considering they are the cornerstone for a good SASUI development process, in Chapter 6 the methodology and architecture defined in this thesis is scrutinized regarding each demon.

2.3.2 Situation Awareness-Oriented Design (SAOD)

The knowledge of SAW "demons" and the model of SAW led to the proposal of the Situation Awareness-Oriented Design (SAOD) [Endsley12], the most applied design

methodology to support SAW [Endsley15a]. The goal of this methodology is to directly support higher levels of SAW (comprehension and projection) on the UI, reducing unnecessary mental workload from the user to create a mental picture of the situation [Onal13].

SAOD is an iterative process based on three phases: a Goal-Directed Task Analysis (GDTA) to identify information requirements, a user-centered design (assisted by 50 guidelines) and an evaluation round. After evaluation, design can be restarted to be improved and reevaluated.

2.3.2.1 Goal-Directed Task Analysis (GDTA)

GDTA is a knowledge elicitation technique, considered a Cognitive Task Analysis (CTA) [Endsley00b]. However, since a CTA must have three parts (elicitation, representation and analysis) [Crandall13], and GDTA is elicitation-based, it can only be considered as a piece of a CTA (and not the complete process).

GDTAs are also different from other CTAs, as they do not focus on system constraints, but rather gives a technology free view of the decisions to be made [Endsley13]. They can be used to define goal and data-driven processes in UIs, but since they do not define a clear sequence of actions of procedures, they do not substitute some other task analysis, such as the Hierarchical Task Analysis.

Discussions apart, GDTA is the most established technique to assist defining SAW requirements to this day. Its execution is described in [Endsley12] and Section 5.3.1.1.

2.3.2.2 Guidelines

Endsley and Jones presented a list of 50 design principles that can be used by developers and designers to ensure UIs support and enhance users SAW when designing for complex systems [Endsley12]. These guidelines are grouped in six categories:

- **General:** generic principles to determine information presentation in a system. Presentation can influence SAW by determining how fast and accurate information can be acquired;
- **Confidence and Uncertainty:** principles for managing data uncertainty in decision-making through enhanced visualization techniques;
- **Dealing with Complexity:** principles for handling display and task complexity, by grouping information, reducing density and others;

- Alarms, Diagnosis and SA: principles for dealing with alarms, considering workload against compliance (trust in automation);
- Automation and SA: principles for dealing with out-of-the-loop syndrome and trust in automation, such as providing support rather than decisions, automating only routine actions and others;
- Supporting SA in Multi-Person Operation: principles for dealing with SAW in teams, by supporting assessment of team members tasks, providing common relevant operating picture.

Some of the guidelines, but not all, are presented below [Endsley12]:

- Organize information around goals and support goal-driven processing: goal-oriented information displays should group information necessary to take a decision, providing a layout that facilitate for users locating information for each goal;
- Present level 2 information directly: directly presenting level 2 SAW information alleviates cognitive load for users to ratiocinate on raw data (level 1);
- Provide assistance for level 3 SA projections: developing level 3 SAW (projection) is a mentally taxing process, but it can be facilitated by UIs showing trends in situations or information;
- Support Global SAW and trade-offs between goal-driven and data-driven processing: providing an overview of the situation at all times is essential to improving users fast goal switching and holistic view of the situation;
- Make critical cues salient: critical cues for activating mental models through pattern matching should be made evident in the UI;
- Use parallel processing capabilities of users: structuring information among displays to support attention cycling, or using different media to present information (auditory, tactile) can improve users SAW acquisition;
- Use information filtering carefully: information not specified by the Cognitive Task Analysis should be dispensed from the User Interface;
- Manage rampant featurism through prioritization and flexibility: as systems and features continue to grow, users may get confused by data and feature overload. Besides limiting the number of feature to the extreme necessary, managing features occupation of screen (and their salience) and allowing user personalization can be important to allow the collection of features to grow without confusing users.

These guidelines were chosen to be explained because they are the most relevant for the domain explored in this thesis. Many works and industry systems used these guidelines [Bolstad06, Onal13, Botega15, Opto2213].

To conclude SAOD methodology, SAW evaluation methods are presented in the next section.

2.3.3 SAW Evaluation

The last stage of SAOD is an evaluation round to assure the quality of the product generated.

Several techniques to evaluate SAW have been discussed in the literature and Salmon et al. and Endsley and Jones listed many [Salmon07, Endsley12].

Different techniques arose because of different necessities of SAW in domains, different influencers of SAW and different decision-making conditions. Table 2.1 synthesizes some of these techniques.

Five main characteristics were determined to organize the SAW measurement techniques:

- Abstraction of SAW measurement {indirect, direct}: techniques can infer a direct value of SAW, or a related value that probably is influenced by SAW;
- Category {process, subjective, objective}: techniques can either measure a process, or give a subjective or objective measurement of SAW;
- Real time {yes, no}: some techniques are able to measure SAW in real-time, such as Corsage [Bin14];
- Measurement time {Interrupt, post-execution, during execution}: measurement can occur during the system execution, after the system/simulation execution, or the user can be interrupted to be analyzed;
- Analysis method {computed, observer, user}: techniques analyze SAW by computing automatically the results, or by an observer analysis, or by the user him/herself rating their SAW.

Table 2.1: Comparison of SAW measurement techniques.

Technique Name	Abstraction of SAW measurement {indirect, direct}	Category {process, subjective, objective, behavior}	Real time {yes, no}	Measurement time {Interrupt, post-execution, during execution}	Analysis method {computed, observer, user}
SAGAT	Direct	Objective	No	Interrupt	Computed
SPAM	Direct	Objective	No	during execution	Observer + user
SABARS	Direct	Subjective	No	during execution	Observer
SART	Direct	Subjective	No	post-execution	User
Performance	Indirect	Objective	No	during execution	Observer or computed
VPA	Indirect	Process	No	during execution	Observer + user
Usability	Indirect	Subjective	No	post-execution	User
Corsage	Direct	Objective	Yes	during execution	Computed

The first technique from Table 2.1 is SAGAT (Situation Awareness Global Assessment Technique) [Endsley95b]. This technique is based on interrupting the user to ask questions related to the three levels of SAW. Interruptions have to be randomized, so that the user does not know when it will happen and cannot prepare for it. Questions are based on the GDTA created for the system.

SPAM (Situation Present Assessment Method) is an online probe technique, in which an observer asks questions for the user during the task, and the answer and latency to answer is analyzed as a measure of SAW [Durso06].

SABARS (Situation Awareness Behavioral Rating Scale) is a technique that consists of an observer (an SME) watching the user during a task and giving a score for each user based on observable behaviors and decisions [Matthews02].

SART (Situation Awareness Rating Technique) is a subjective technique in which users rate their own SAW after the task [Taylor90]. It was originally developed for pilots, but was acknowledged to be accessible and is used in several different domains [Bashiri14].

Another type of technique is the performance measurement. This is highly dependent on the domain and the goals of the user, but analyzing performance of a user task can indicate that he/she possess some level of SAW, therefore this is an indirect level of assessment. Some authors argue against the use of performance (or at least against the use of **only** performance as a mean to measure SAW) [Endsley15b], while some other authors argue that performance should actually be the main artefact for measuring SAW [Dekker04, Sidney10].

Another form of measurement is by mapping and analyzing the process that users go through to develop SAW during a task, such as psychophysiological metrics (such as eye movement, electroencephalogram, electrocardiogram). This inference of cognitive processes from physical reactions is an indirect and not widely used method for SAW evaluation. Approaching a similar line is the Verbal Protocol Analysis, used to obtain insights about cognitive aspects of complex behaviors [Smolensky93].

Finally, usability evaluation of UIs are also used to indirectly determine SAW levels, such as used by Onal et al. [Onal13].

In this thesis a mixture of usability, performance, and SART were used to evaluate the user SAW with different UIs. This evaluation is presented in Chapter 6. Appendix D presents the questionnaire used.

SART and SAGAT are the most used forms os SAW evaluation [Salmon07]. We adopted SART because of its faster use time, its standard metrics, since it is always the same questionnaire independent of domain and tasks possibility, while SAGAT is particular to each task, and because a testable response mechanism was already used to identify good or bad decisions (overlapping with SAGAT).

2.4 Industrial Maintenance

The development of this thesis was guided by a study case and an in-depth analysis of an activity within industry, maintenance. This activity was chosen due to its importance in the industry, its presence in every plant, its complexity and requirements for safety and efficiency (to reduce downtime), and also for its abundant literature.

Maintenance, also named Maintenance, Repair and Operations (MRO), is important to any business activity, because it aims to preserve the availability of the assets used by the

business. The maintenance plan usually focuses on diminishing the cost of keeping the business operational, while maintaining the equipment.

Maintenance today became essential for the sustainable and continuous development of industry and society. Aspects like environmental, economical, safety and energy saving are influenced by maintenance. Furthermore, it is an important activity for improving enterprises and their competitiveness.

For a long time, maintenance was improved through enhancements in its processes. Currently, new Technologies are being introduced to improve maintenance, such as E-Maintenance, Internet of Things and Wireless Sensor Networks for data acquisition and management, and Augmented Reality (AR) and Head-Mounted Displays (HMD) for visualization [Oliveira13].

Complementing visualization technologies, there are studies about User Interfaces in maintenance. Considering UIs for E-maintenance systems, three studies laid the ground for research [Tretten11, Wandt12, Tretten14]. Their initial studies listed usability problems in current Computerized Maintenance Management Systems (CMMS) and found that major points of improvement are context-awareness (applications aware of what is happening for the user and adapting to it) and data-related improvements (facilitate data input, lack of data compatibility between systems and data multiplicity).

Additionally, for maintenance, the focus of UIs should not be only on efficiency, but also on efficacy, and related to this last one is safety. Overall, in industrial environments, safety (for both users and equipment) is the most important requirement [Ward08], because of the risks of the tasks and the gravity of consequences of accidents [Dhillon06].

Considering accidents, Rasmussen classified human errors and malfunctions in a taxonomy [Rasmussen82]. In this model, it was recognized the impact of cognition, with elements such as information acquisition, situation assessment and awareness and decision-making. All of these were judged as critical in maintenance, by Bouyer and Sznalwar [Bouyer05], Oedewald and Reiman [Oedewald03], Antonovsky et al. [Antonovsky14], Bullemer and Reising [Bullemer13], and many others [Golightly13, Endsley00a, Chinoy11, Sneddon13, Nazir12, Dhillon06].

Since safety is important, and as SAW was identified as an important aspect in safety and accidents in industrial environments, the next section discusses SAW in industries.

Situation Awareness structural division in maintenance

Situation Awareness in maintenance and industrial work differs from the traditional area of SAW application, Command & Control, because of the strong procedural element in this scenario, which can contrast to the situational nature of SAW. As stated by Sian et al. [Sian96]:

“The maintenance environment, though hectic, changes slowly relative to flight operations (see discussion of human error). In terms of SAW, AMTs (Aviation Maintenance Technicians) must have the ability to extrapolate the consequences of one’s errors over hours, days, and even weeks. To do this, the situation awareness cues that are taught must be tailored to fit the AMT environment using MRM-specific (Maintenance Resource Management) simulations.”

Sian sentence shows that SAW is important for maintenance technicians, but must be considered different from Crew Resource Management (aircraft crew) SAW. From their point of view this difference is crucial to create adequate training of a MRM program, as densely explored by Robertson [Robertson98]. From this thesis point of view, this difference will differentiate UI development to support SAW in maintenance work (as opposed to C2).

Poor SAW is an acknowledged source of errors in industry and maintenance [Dhillon06] and whenever reliability and resilience are important, such as it is in maintenance, there must be a focus on being responsive to situations [Golightly13]. The conclusion by many researchers is that user’s SAW, when using technological apparatus, is directly correlated to the UI [Endsley12].

Coherently to these assertions, in order to improve SAW in industry through User Interfaces, static information/context must be considered by the SAW-driven design process, regardless of their non-situational nature, because of their effect on users’ perception of their situation.

For instance, even though the risks of maintaining and testing the speed regulator in a power generator unity are always the same (electric shock, height fall, high pressure, oil leaking), keeping them in the user working memory (short-term) through UIs may reinforce the mindset to assure a safe and effective work. Moreover, such information will also assist the user’s decision-making process by providing comprehensive context for an analysis of the situation, in other words, they improve the speed at which users develop their SAW, because these risks are part of the user current situation.

To be clear, as analyzed in Section 2.2.2, SAW is not static knowledge. However, static knowledge does affect SAW and the decision-making process, especially in maintenance work, where procedure knowledge is the center of work. A UI that disregard this fact, will be quickly dismissed by workers. Therefore, focusing only on situational information in UIs for maintenance work is not enough (or any other work domain).

Besides the implications of displaying static and non-static information, a comprehensive design for SAW must consider that a system/domain may contain multiple aspects (or types) of Situation Awareness, as stated by Golightly et al., or awareness itself could be composed of several factors, as identified by Carrol et al. [Golightly13, Carroll03].

Pew [Pew00] (reinforced by [Bossé07]) even defined the aspects of SAW as (these definitions are followed by the names used in this thesis, when different):

- Spatial awareness (Environment Awareness)
- Mission/goal awareness (Procedure Awareness)
- System awareness
- Resources awareness (Equipment Awareness)
- Crew awareness (Team Awareness)

Regarding specifically maintenance, the works of Endsley and Robertson and Golightly et al. identified four elements as the basis of SAW in maintenance field work [Endsley00a, Golightly13]: understanding the equipment, to identify problems (diagnosis) and predict failures (prognosis); maintaining team synchrony, to collaborate and coordinate tasks to achieve a common goal; comprehending the environment and their risks, to avoid accidents; having a good corporate environment, to have a standardized work routine and terminology and good communications with other areas for shift scheduling and provisioning of assets. The majority of these factors are also present in any industrial work, and most are present in any type of work (C2 included).

Sneddon et al. exposed that the most significant impact in SAW is caused by stress, and not fatigue/workload and sleep disruption [Sneddon13]. They also proved that Situation Awareness is associated with safety indicators, such as unsafe behavior, near misses and accident history. Likewise, Bullemer and Reising analyzed major incidents from industry, discovering that 50% involved failures associated with SAW [Bullemer13]. According to them, Team SAW could be improved with better communication, supervision, pre-job briefing and User Interfaces.

Improving upon literature for SAW in maintenance and industrial work, combined with on-site analysis, this thesis uses the strategy of structuring Situation Awareness around seven aspects, each one of those with its respective requirements and design strategies. The framework to enable this is presented in Chapter 4.

To complement the concepts of maintenance and Situation Awareness, before the state of the art is presented, a review of Model-Based Design and End-User Development is presented in the next section.

2.5 End-User Development and Model-Based Design

End-user development (EUD) allows users acting as non professional software developers to create, modify or extend a software artefact [Lieberman06]. Usually, users have access to an application, but cannot augment or change it. In EUD, through a set of tools or techniques, users could modify behavior or User Interface or even create new functionalities that were never foreseen by the programmers [Paterno13].

As an example, Power Point could be considered an EUD tool. If it were only considered the creation of content, Power Point would not be classified as EUD by most authors. However, if considered the possibility to create complex animation (simulating logical expressions – IF (AND, OR)), the possibility to program buttons (and program macros), and even the functionalities to link presentation with spreadsheet/databases (Excel and Access), Power Point is a good example of the power of an intuitive EUD.

Most examples of EUD are spreadsheets and databases, because they are heavily used in the corporate market and have capabilities to easily create logical conditions, macros and scripts that can extend them.

Nonetheless many current tools can be considered EUD. Game Engines are another good example of EUD, since they abstract a huge amount of artefacts, allowing game developers to work on high-level structure. Many engines these days allow game developers to develop without any coding, while most engines are script based.

There are many approaches to creating an EUD tool or technique, such as programming by example, visual programming, macros and scripting. For this thesis, the chosen approach was of model-based development, in which the user uses high-level

(abstract) models to define the system functions and these models generate the application [Paterno01].

Model-based UI Design (MBDUI) assists managing complexity, decreasing the effort to produce UIs but at the same time ensuring quality [Meixner14]. They allow designers to work on a semantic level, without the concern for implementation details.

The Cameleon Reference Framework (CRF) was the first reference framework for MBDUI independent of the UI domain [Calgary03]. It is widely adopted in the HCI community to assist authors of MBD processes to classify and organize their models [Meixner14].

CRF is also relatable to the MDA framework, proposed by the OMG [Kriouile13]. Both are based on the concept of Computation Independent Models (CIM), Platform Independent Models (PIM) and Platform Specific Models (PSM).

The concepts of CIM, PIM and PSM are [Truyen06]:

- CIM: it is usually a task or domain model, which uses vocabulary from the SME universe. It presents what the system is supposed to do, but hiding any implementation detail;
- PIM: presents a set of services, components or other form of defining the system, but in a level high enough that it can then be mapped to one or more platforms;
- PSM: includes information about specific technologies to implement the PIM in a system.

2.6 Final Considerations

This chapter presented the concepts of SAW, maintenance and MBD. Analyzing the Human Factor literature, Situation Awareness drew attention for its proved importance in Command and Control domains, for the lack of studies of SAW support UIs for industry, and for the accidents and errors appointed as caused by insufficient or erroneous SAW in many industrial operations.

In a desire to implement a Cognitive Engineering approach to UI design for maintenance work, SAW was the cognitive model chosen to analyze the decision-making requirements of this domain.

This chapter also presented a structural view of SAW in industry and maintenance, to justify the necessity to better systematize SAW in these domains, an opinion already shared by authors in the literature, although never discussed in such a relentless manner.

Then it presented the most used and one of the few approaches to support SAW UI design (SASUI), the Situation-Awareness Oriented Design (SAOD) methodology. SAOD has three steps: a Goal-Directed Task Analysis (GDTA); 50 design guidelines; and a SAW evaluation.

Nevertheless, while SAOD is a good start, it does not systematize the development process. Because of its high-level of abstraction, it can be applied to any domain, but at the same time it lacks a more rigid structure, or a better defined process, to not only speed up development time, but to also standardize results. SAOD also does not provide any tools for designers to work on.

Therefore, considering SAOD drawback, this thesis proposes a model-based process to structure SAW-driven UI design, to allow the creation of tools that assist designers and can speed up development, and to transform the abstract SAOD into a software engineering process, making it more applicable for industries.

To implement SAOD into a model-based process with supporting tools, the first step was an analysis of UI requirements and current UIs for maintenance. From this analysis (and the literature review of SAW), it was clear that a powerful model-based UI needed not only context-awareness, but also a model of Situation Assessment.

Therefore, in the next Chapter the state of the art is presented, with discussions of UIs for maintenance, Situation Awareness Support System (SASS) development strategies (including models of Situation Assessment) and model-based strategies that substantiated the solution proposed in this thesis.

Chapter 3

STATE OF THE ART

3.1 Initial Considerations

First and foremost, this thesis is proposing a methodology for design and development of SASUIs for maintenance technical works in industry. Therefore the initial discussion is regarding Situation Awareness Support UI and System development. For this section, first models of Situation Assessment (SA) preeminent from the literature are presented. As explained in Section 2.2.1, SA is the process of gathering and interpreting information to form SAW. Since a SAW model is a model of a cognitive process (like the Endsley's model), the term computational (prescriptive) SA model (sometimes termed as Situation Analysis) is used to describe a model to process and provide data and information in decision support systems to improve its support for SAW.

From a subset of models of SA, multiagents as SA models are also presented. Then, a repertoire of SAW research offering methods and guidelines for SASUI design are explored.

Additionally, to explore the weakness of current SASUI design methods, Model-based UI development methods are discussed. Nevertheless, since this thesis proposes a model of MBD with agents, only Model-based UI development for agents is discussed. Finally, we present a discussion of current UIs for maintenance.

The chapter is concluded with a discussion to compare the current state of art of SAW support UI design with this thesis, with an analysis of SAW quality indicators for several works.

3.2 Situation Awareness Support System and UI development

Although the only comprehensive methodology proposed for SAW-driven system design is the Situation-Awareness Oriented Design (SAOD), many others works have proposed solutions for punctual problems. Some of these works tried to model the SA process and use it to build improved Situation Awareness Support System (SASS), and some focused on giving design guidelines for improved SASUI. They will be reviewed in the following sections.

3.2.1 Models of Situation Assessment

There are many attempts to model Situation Assessment (SA) or Situation Analysis computationally, but unlike the cognitive SAW model (Endsley three levels model of SAW), no computational model of SA has been dominant or widely adopted.

The first model reviewed is called Situation Awareness Fuzzy Cognitive Mapping (SA-FCM) [Jones10]. This model represents an actionable model of SA that is designed to mimic effective decision-making; in other words, it represents a computational naturalistic decision-making model [Jones10].

In this model, a FCM is designed from the GDTA, and then during run-time this FCM is fed with the inputs to generate the optimal solution for users [Jones9]. During run-time it is possible to visualize the explicit calculation of levels 2 and 3, which helps users achieve trust in the automation (combating complacency) [Jones11b].

However, the solution was not completely validated, with some experimental works showing that it increases too much the level of complacency in its automation, damaging performance in the long run [Jones15].

Jones et al. presented another solution for SA modeling, in the form of SAW agents [Jones11a]. They proposed three guidelines for future studies regarding SAW agents, which were used to propose the autonomous SAW-driven interface agents from this thesis (Section 4.4).

One of the criticisms regarding the SA-FCM and the SAW agent's model is the lack of an ontology. That is why Kokar and Endsley, Matheus et al., and others proposed the use of ontologies to increment SA-FCM, allowing the reasoning to include semantic knowledge [Matheus03, Kokar12].

On the line of SAW agents, Hoogendoorn et al. proposed a SA model for agents that models the human mental model using beliefs [Hoogendoorn11b], with the goal of making agents aware of the situation. They created a cognitive framework for agents, based on the evaluation of simple beliefs that can be combined to form complex beliefs, and using time to evaluate future beliefs (level 3 SAW). This project was later improved to aggregate a model of Functional State of humans, which calculates the impact of fatigue/stress in SAW development and decision-making [Hoogendoorn11a].

Bosse et al. improves upon the work of Hoogendoorn et al., by proposing to also model SA using a Temporal Tracing Language (TTL), incorporating qualitative time references [Bosse12].

Another work using inference and TTL is from Baader et al. [Baader09]. They proposed a system architecture that allows automated reasoning by formal logic, which can be consulted by the user. Their support for a declarative approach to build a SAW is similar to this thesis.

Feng et al. presented a situation model for shared SAW that is also context-aware [Feng09]. Entity agents interpret the current context and situation through a rule-based engine and provide event classification, action recommendation and proactive decision-making, to inform the user the best decision through a UI. However, their levels 2 and 3 SAW model is dependent on pre-made functions (hardcoded) for their established domain (military).

Naderpour et al. focused not only on a SA model, but also in the implementation of a SASS [Naderpour14]. The main objective of their study was to develop a cognition-driven Decision Support System for use in abnormal situations. The proposed SASS implements two contributions: 1) a process to model abnormal and hazardous situations; 2) a dynamic Bayesian network to support operators in acquiring SAW, with a fuzzy risk analysis to estimate risk for the operator (first introduced in [Naderpour12]).

Nevertheless, this model has a heavy focus on safety in command and control, while maintenance technicians (and any other field operator) would have different risks. It also focuses only on risky and abnormal situations, thus tending to increase the worker safety but not his/her efficacy and efficiency. It goes to the point that it classifies situations by only two types: safe or hazardous.

Finally, some works focus on machine learning, like Lu et al., which proposed the use of fuzzy least squares support vector machine technique to analyze related data sources, to maximize the learning of past successful decisions [Lu08].

While their solution (and machine learning overall) is interesting and could be studied to be included in our SA model (to improve situation classification and recognition of entities represented in the UI), machine learning can be complicated for industrial work (such as maintenance). As stated by Naderpour et al., the use of machine learning in real cases, even for command and control scenarios can prove challenging because of the difficulty for data acquisition¹ [Naderpour14].

Overall, a significant difference in goal of the literature about SA modeling, and the model of SA which is proposed in this thesis, is that most papers focus on completely modeling SA to automate decisions or recommendations, as end goal. For this thesis, the focus is not to automate or even suggest course of actions (decide for the user), but to provide information for the user to make the decision. Therefore the level of intelligence and inference necessary, when compared to [Naderpour14] or [Hoogendoorn11b], is not the same.

While they focus on identifying situations for the user, this thesis has a focus on comprehending how situations can influence user UI and how UIs should support users in better understanding situations.

This can be both considered an advantage and disadvantage of this thesis. Advantage because it allows for faster and clearer (for SMEs) construction of SA models, which results in quicker results (the UI) and easier creation (and control) of new UIs. As stated by Yim et al. [Yim13], the human mental model is too complex to be defined as an algorithm, but most experts can use production rules (IF-THEN structures) to define their knowledge.

This sentence from Feng et al. reinforces the importance of production rules [Feng09]:

“Being one of the established ways to implement situation awareness, production rules provide us with a simple and comprehensible specification for recording the heuristics articulated by the domain experts.”

The work of Feng et al. has several similarities with this thesis. Their focus on SAW and context-awareness, by the use of agents, is similar to the solution proposed here. Even the inference rules (from their DROOL engine) are close to the SA modeling proposed. However, the differences are in the focus to design UI, and not only model SA, and the multiagent solution (Chapter 4) is in the end (albeit similar at high-level) different and more UI focused than what they proposed.

¹ Converting a dialogue record [Benyon14 page 395] to a decision/knowledge record.

One disadvantage is the lack of powerful inference mechanisms, which could implement SAW, especially level 3 SAW (projection), more significantly. However, it is a consensus in the SAW community that no solution so far has implemented a proven effective and generic way to calculate projections of the future [Jones11a, Jones11b]. Nonetheless, regarding maintenance, some of the most complex level 3 SAW in this area would be related to predicting equipment behavior, and in this case the solution would not be to force SMEs to model this level, but to use knowledge and functions that are already present in the E-Maintenance systems to obtain such answers about equipment future condition (such as Condition Based Maintenance).

Therefore, for the domain chosen (technical work in maintenance), the disadvantage will not downgrade the UI/system performance, and including more sophisticated solutions (such as [Bosse12] or even [Jones11b]), would decrease cost-benefit, for the effort it takes to initially model all the necessary requirements (Belief-Desire-Intention, or ontologies, or Fuzzy Cognitive Mappings, or others).

At the same time, due to the componentization of the architecture, each agent has their own SA model. This opens the possibility to create improved and more complex SA models if enough time is available, which allows for the use of, for instance, SA-FCM, in our architecture. These improved SA models are explained in Section 5.3.2.

Thus, the recently reviewed concepts that are used in this thesis are: a situation model, the FSA (which can also be the start of an ontology) [Feng09]; a declarative technique to specify formal logic (production rules), but using predicate logic instead of description logic [Baader09]; computational agents to model mental models [Jones11a, Hoogendoorn11b]; structuring agents SA according to experts [Jones10].

These computational agents are another interesting and related attempt to model SA. They sometimes deal more effectively and naturally with SAW in system design [Bossé07]. The next section explores the concept of computational agents for SAW.

3.2.2 SAW and computational agents

Although the definition of autonomous intelligent computational agents (henceforth will be called only agents) is not universal, the definition by Wooldridge is adopted in this thesis [Wooldridge09]:

“An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.”

Basically, an agent is an independent element that receives an input, and processes it autonomously to produce an output.

One work already explored in the previous section, but that also is based on agents is [Feng09]. As already commented, their strategy revolves around a group of entity agents, one for each user, that communicate with a situation model to classify events and recommend actions.

Others works are based on Belief-Desire-Intention (BDI) agents, such as Urlings et al. [Urlings03] proposal to use BDI agents for military environments (simulated in a game), or So and Sonenberg rule-based knowledge representation and forward reasoning to provide proactive behavior in agents [So04].

The work of Mostafa et al. focus on enabling multiagent systems with SA, to improve the decision-making capacity of the agent [Mostafa13, Mostafa14]. The solution is a Situation Assessment model that provides a situation ontology to comprehend situations and assess agent's performance. However, in this scenario, the human operator functions only as a supervisor.

Some of the works proposed innovative solutions to model Shared SAW, such as Blom and Sharpanskykh mathematical formalization of shared SAW for multiagents system [Blom15] and Bosse et al. agent based social simulation framework [Bosse13].

Finally, Jones et al. [Jones11a] proposed a model of Agent Situation Awareness, in which agents can understand not only the situation, but also the necessary decisions users will take, instead of only being context-aware. This can provide for a more synergistic collaborative relationship with agents and users. Most current agents lack the ability to generate higher level cognitive process (level 2/3 of SAW), offering only perception (level 1 SAW), usually through information (sensor) discovery and visualization.

Jones et al. proposed a framework with the following design considerations [Jones11a]:

1. Agents should support human decision-making:
 - a. What should be done: agents that want to support decision-making need to adhere to a theory (in this thesis Situation Awareness) and implement it to support users, while also having a user centric approach to their design.
 - b. What most do: most agent-based solutions are techno-centric.

2. Agents should be designed with the lowest level of automation needed:
 - a. What should be done: agents that are too autonomous leave users out of the decision loop, hurting SAW development, which can result in the wrong decision by the operator if he/she needs to interfere in the system.
 - b. What most do: many works on agent SAW focus on providing SA for the agents [Mostafa13], with the human being a monitor to assist these agents acquiring the correct SAW.
3. Agents should provide support for anticipatory and predictive awareness:
 - a. What should be done: agents for SAW should map level 2 (comprehension) and level 3 (projection) of SAW and use it to support users' development of SAW.
 - b. What most do: most works on agents focus on the data combination process to integrate information and evaluate information uncertainty [Hoogendoorn11a].

Analyzing the literature on computational agents for SAW, one sentence from Mostafa et al. makes clear the goal of most of the community:

“...an independent mechanism that exploits this characteristic in the multiagent systems design can lead to emerging SA in agents [5]. The challenge is to formulate an efficient approach of SA to enhance agents' autonomic actions.”

The focus of most is clear on empowering agents to have better autonomic reactions for the situation. Thus, for many, SAW acts as a lever to improve context-awareness.

While this is indeed important, the view of agents adopted in this thesis is more comparable to Jones et al. [Jones11a], which request less automation from agents and better assistance to the human decision-making process.

That is why this thesis proposes a model of agents merging UI and SAW, to work in favor of assisting the user during decision, and not assists the agent decision/action only.

Finally, simply increasing automation may incur in problems with trust (and complacency) [Parasuraman08] and out-of-the-loop syndrome (a SAW demon) [Endsley12]. Some works that abused automation had negative results [Jones15], and even authors in High-level Fusion community list the necessity for combining computing power with human cognition as an important challenge [Blasch12].

Therefore, the path chosen for this thesis, to merge intelligence and UI using the agents, seems valid and desired by the SAW community.

Other works that attacks open challenges of SAW from several different angles are discussed in the next section.

3.2.3 SAW guided development and guidelines

Besides computational SA models, many other works were proposed in the literature to deal with challenges in developing a SASUI or SASS.

Some works proved the validity and usefulness of SAOD applying it to create UIs for their domain/problem, such as smart grid [Romero-Gómez13], emergency response [Botega15], HMI [Opto2213], military [Onwubiko12] and many others [Bolstad06, Onal13].

Nevertheless, some proposed new (or sometimes similar) guidelines, as compared to SAOD.

Chinoy and Fischer appointed the need to improve support for SAW in UIs [Chinoy11]. They listed several design and system guidelines, such as “Show radar coverage especially in light” and “Information tailored to user need”, with a focus on a net-centric approach that integrates several databases.

A subject not covered by SAOD is interruptions in ongoing activities, such as when a teammate talks with the worker (which loses focus on the situation). John and Smallman created four design principles for UIs to assist in interruption recovery [John08]: 1) UIs should provide an indicative of changes automatically; 2) UIs must use unobtrusive notifications; 3) UIs should provide summary descriptions of each change, allowing users to choose priorities; 4) For busy displays, UIs should make information about changes be accessed only on demand, to avoid clutter.

Another interesting finding is from Patrick and Morgan [Patrick10]. Despite many design suggestions stating that facilitating access to information is pivotal to SAW, they proposed to increase the cost of access to information when situation is not emergency. By their studies, they concluded that raising the cost of accessing information makes the user adopt a strategy to memorize it. This may facilitate users to develop their SAW, as compared to a UI that allow a cost free access to any information, because that usually leads to the user not memorizing any data.

Concerning models, Villaren et al. were the first and apparently only to propose a model-based approach to SASUI design. Their methodology formalizes operators tasks to lessen the impact on the operator’s SAW at each task transition [Villaren12]. They used a Hierarchical Task Analysis (HTA) linked with a DSA graph, and each task has its graph compared to the next task to spot differences. A UI then supports users in transitioning their SAW internal model between each task.

Nwiabu et al. [Nwiabu12] discussed a framework for UI design that uses HTA, scenarios and requirement analysis to create a SASUI. The interface is based on the results of a HTA, which decomposes complex scenario objectives into small tasks. The UI is capable of automatic reconfiguration to adapt itself to the current context. However, their understanding of a SASUI seems to be only of a context-aware system.

On an alternative route, Stanton et al. [Stanton13] proposes a methodology for system design, targeted at their theory of SAW (DSA), named Event Analysis of Systemic Teamwork (EAST) and developed to examine the work of distributed teams of people in complex socio-technical systems. EAST propose the creation of three interconnected networks: 1) a task network that formalizes task structure, actors and space-time; 2) a social network to map communication between individual within each task, considering location and technologies; 3) a propositional network, which is a knowledge network during each task, clarifying decision-making at each time and SAW requirements.

Overall, the major limitations of current solutions are a lack of MBD or another solution to accelerate UI design and a lack of EUD principles to allow end-users to change the system. Most UI guidelines and techniques consider system development as a one-time activity that should be done before anyone uses the system. While this may pose true for C2, in maintenance work the tasks are always changing, evolving or readjusting for the necessities. A Work Instruction is always a “work in progress” [Carvalho15].

Therefore a method that creates a system and leaves it to be used (but not expand) is prone to be outdated quickly. This view of the constant evolving world of maintenance work is thoroughly analyzed by Carvalho and Menegon, using the concepts of different unpredictabilities in the industry [Carvalho15]. Thus, not only current solutions may take a long time to provide a result (a system and a UI), but as soon as they are ready, they may already be outdated.

To solve this problem this thesis proposes the use of Model-Based Design and End-User Development, with computational agents, for maintenance work UIs that support SAW. In the next section approaches on MBD for agents are reviewed.

3.3 Model-based UI development for agents

Model-based design, model-driven design and model-based engineering are adopted in several domains in the computer science field. For this thesis, because of the focus on SAW and agents, both to represent UI and SAW, only works related to the model-based design of agents are explored.

The work of Eisenstein and Rich was the first to propose the use of MBD to facilitate creating conversational interface agents [Eisenstein02]. By applying a task model to their GUI editor VAMPIRE, designers could specify the GUI and generate a collaborative interface agent for the same task.

Ahrndt et al. were among the first to introduce an MBDUI process for a multiagent system (MAS) [Ahrndt12a, Ahrndt12b]. In their scenario of Ambient Assisted Living (AAL), they argued about the natural fit of MAS for the distribution and autonomy of responsibilities, considering AALs are ubiquitous scenarios. This fact, added with requirements for multi-modal interactions (especially for their elderly audience), raised the point that an MBDUI process can facilitate development by creating UIs to communicate to the agents. However, agents, which are already present in the environment, have a problem of communication with MBUIs. To solve this problem they developed the Human Agent Interface (HAI), which acts as a gateway to deliver messages from UIs to agents and vice versa.

On the concept of agents to generate and control UI, Viano et al. worked on a solution called Auto-Adaptive Multimedia Interface (AAMI) architecture, based on Intelligent Agent collaboration [Viano00]. They used the power of a MAS to generate and adapt UIs during run-time.

Another relevant work coupling MBD and MAS is from Tran et al., which presented a MBDUI process based on MAS (agent-based framework) to generate UIs and applications for databases access [Tran09]. Agents are used to transform task, context and domain models into code.

So far, three different but complementary solutions were explored: MBD to create UI agents, MBD to create UI **for** agents (in a MAS) and MAS to create UIs. All of these solutions are merged in this thesis, since the agents generate the UI and a MBD process is used to model the agents.

On another line of research, there is the work of Ricci et al. regarding agents and Mirror Worlds² (MW) [Ricci15], proposing the use of a high-level abstraction (an agent-oriented programming models) to design and program a MW.

Exploring the path of EUD, there is Pantagruel, which has a high-level visual programming language that allows users to define orchestration logic of pervasive environment through rules created upon entity classes [Drey09, Drey10, Drey12]. The visual language uses a taxonomy-based approach, facilitating reuses for any domain, while allowing users to create filters and rules to control sensors and actuators (entities). Entities can even be addressed as classes (every entity of a type) or as specific entities. Their defined entities are similar to the defined agents of this thesis (in particular equipment agents).

Continuing with EUD, component-based architectures provide a gentle slope to end-users in understanding and designing systems using components. The FreEvolve architecture allows tailorability by end-users, as it instantiates at run-time copies of pre-defined components from a server [Wulf08].

These components are compared to the agents defined in this thesis, and FreEvolve was used as inspiration for the architecture proposed for this thesis in Chapter 4. In fact, many interesting solutions from the literature inspired the agents and MBDUI process proposed in this thesis.

Initially, Vianco et al. [Vianco00] inspired the agent solution proposed in this thesis (Chapter 4). Their use of agents that consider information about the work process, the environment, human factors guidelines, HCI guidelines and the operator to generate (and adapt) a UI, fit with the requirements from SAW in maintenance and industrial work domains.

Ahrndt et al. [Ahrndt12a] proposal of MBDUI comes similar, but, contrary to their solution, in this thesis, agents are modeled as abstraction of real world entities (equipment, environments, etc), and assembled ground up to become a UI. Therefore they do not require HAI, because UI and agents are the same (with agents actually generating the UI).

The idea of a MW, from Ricci et al., aligns even better with what is proposed in this thesis [Ricci15]. The following sentence from Ricci demonstrates the resemblances:

“On the background of MWs there is the broad idea of using agent-oriented abstractions to shape the continuous real-time distributed flows of situated information generated by the physical and

² “digital world shaped in terms of a multiagent system, situated into some virtual environment, which is coupled to some physical environment” [Ricci15]

social layers, as well as of the distributed intelligent software processes that work on that information in order to provide some smart service or functionality”

Therefore, they propose the use of agents as an abstraction to represent real world entities. These agents can represent entities from the physical, social or even software layer. They go as far as to propose that humans visualize and interact with agents (entities) by the use of a “user assistant mirror agents”. This system of “assistant to mirror agents” is exactly the way the agents proposed for this thesis work.

The agents proposed in this thesis **are** the UI and mirrors of real world entities. Instead of being agents in the environment to act in the real world, they act (generating, controlling, tailoring) on the UI through observation of their counterpart entity. Their characteristics will be defined in the next Chapter (Section 4.4).

Nevertheless, as the Internet of things (IoT) advances and MAS are applied to the industries, HAI [Ahrndt12a] could facilitate communication from real world agents (IoT agents) to the UI agents. This communication is currently abstracted as communication with a system of system (E-Maintenance system for the maintenance domain).

Furthermore, as explored in the literature, the importance of a user-centered design, especially considering user participation in the process, is vital to generate adequate and usable systems [Paelke15]. EUD can be considered a great contributor to user-centered design, as the user him/herself design and creates (or tailors) the application [Wulf08].

Pantagruel is an EUD tool that allows end-users to actually design their own systems functionalities [Drey12]. A difference between Pantagruel and this thesis is that they have also have actuators as entities (our agents). This is important to them because they are aiming to control a pervasive environment (using actuators), and control is the key word here. In this thesis, there is no aim at controlling the environment at this moment of the implementation, but the control aspect was already specified (for future uses), which would be the System Awareness, the awareness that allow to understand the underlying system and automation and possibly control this system.

Closing the cycle, to support EUD, MBD (or Model-Driven Engineering) is usually adopted. The benefits of adopting MDE, as listed by Dubois et al., are not only a great match for EUD, but a great match for what is required from this thesis [Dubois14].

Before presenting an in-depth discussion of related works, the next section introduces current UIs for maintenance.

3.4 User Interface for maintenance

One of the initial works to define UIs for maintenance is the technical report of Wampler et al. [Wampler93]. In their work, they specified how a UI should be developed for an Integrated Maintenance Information System (IMIS). Wampler and colleagues defined how the UI should look like (for desktops and handhelds) and possible UI components.

Since then, the IMIS evolved into the E-Maintenance system and many projects explored UIs for maintenance technicians and personnel's, while at the same time companies moved towards handheld for maintenance tasks [Jantunen11].

To advance UIs, many projects explored the use of Augmented Reality and Head-Mounted Displays. AR as a visualization solution focus on reducing the cognitive load of understanding virtual information and applying it in the real world [Feiner93, Lipson98, Henderson10, Yim10, Liu12, Westerfield12], while also combining with current systems for maintenance support [Friedrich02, Stock05, Benbelkacem11, Espindola13].

On a similar line, works regarding HMDs focus on providing optical see-through devices with high usability UIs to integrate with maintenance work [Nicolai06, Asai08, Aleksy14, Yang15, Enblom15].

All of these lines of projects (and others) in maintenance UIs offered improvements, but none of them analyzed SAW and what are the cognitive demands for decision-making in maintenance. Most projects focused either on technology problems, or simply on converting current manuals to a digital format, offering support for procedures, but not for Global SAW (Global and Local SAW will be discussed in Section 4.2). A few also had context-awareness, but small considerations for cognitive demands.

Since a literature review point SAW as a cause of many major accidents and small errors, and as being indispensable for decision-making, a SAW-driven design process is the solution proposed in this thesis and the point of comparison with other solutions for maintenance UIs.

Our approach differs from most studies developed for maintenance technicians (and other field personnel), as these systems are applied in laboratories with ideal conditions, disregarding the user's SAW in real work contexts. This can be misleading, because it ignores the fact that up to 88% of human error is due to problems in SAW (depending on the domain), 60-80% of errors in industry are attributed to humans and 15-20% of these errors involve maintenance [Endsley12, Dhillon06].

No project for UIs in maintenance, as far as the authors know, included a discussion for SAW. A few AR projects (not for maintenance) discussed SAW, but either as a buzz word [Livingston11] or in the C2 domain [Irizarry13]. Yim et al. provided a UI for maintenance that considers cognitive demands (using Ecological UI design), but not SAW [Yim11].

Finally, this quote from Enblom and Eskebaek express how many researchers of HMDs feel [Enblom15]:

“From another perspective it (HMDs) can also have positive effects on safety for workers doing any physical labor. If a PC program is used and the user has moved away from the computer, warnings will not help much. If HMDs were added to other mandatory safety equipment the user would always have a screen located right before the eyes and warnings would be almost impossible to miss.”

It is clear that in the future HMDs have a strong possibility of becoming mandatory safety equipment and when they do, the suggested “warnings” will have to be planned to not be cumbersome and not overload users. This is already part of Situation Awareness focus (alarms), as viewed in Section 2.3.2.2, and is also already addressed in the methodology proposed in this thesis for UI design and development.

To finish this chapter, the next section offers a discussion regarding some works in the state of the art of SASUI design that are directly comparable to this thesis.

3.5 Discussion and comparison of the state of the art of SASUI design

Along this chapter, many works were analyzed and compared to this thesis. However, there is potential to execute a deeper comparison to some projects that have higher resemblance and compatibility of goals.

The works chosen for this comparison are processes to design UIs or systems to support SAW. They need to provide a complete process to design and develop UIs, or they would not be comparable to this thesis. For instance, a solution to a more punctual problem (a model of SA, like [Jones10] or [Hoogendoorn11b]) is not comparable, because this thesis goal is not to provide a model of SA.

Therefore, even though the thesis has a model of SA, comparing exclusively this aspect would not generate a valid discussion, because the decision on which model of SA to adopt (or design process, model-based process, architecture, etc) was made to fit the project and cannot be dissociated with all its parts.

For the discussion and comparison of projects, a set of quality indicators for SAW support was devised. These quality indicators are based on the guidelines from SAOD (the main SAW development methodology to date) and were put together in Table 3.1 to facilitate an analysis of many projects that support SAW development.

Works that expect to accomplish any form of direct SAW assistance or guidance to the user, whether because of UI, system or intelligence enhancements, but at the same time do not actually consider basic guidelines of SAW design, are constantly questioned by many authors in the SAW community [Parasuraman08, Wickens08, Jones11a, Endsley15].

Notice that, although this thesis is focused on the industry domain, the set of quality indicators could be used to discuss and compare any work from any area that is aiming in improving UIs for SAW support (for instance areas such as C2 or aviation, which have abundant literature for SAW).

These indicators are divided in many areas, ranging from correct analysis of cognitive demands, to execution of the guidelines from SAOD.

The first indicator is the used CTA method (like the GTDA), which is considered essential to gather requirements to build a SASUI. If the project has any forms of cognitive, goal or decision driven analysis, they are listed in the table.

Another item is the presence (or not) of a SAW evaluation. As modeled in SAOD (and as considered by any HCI or Cognitive Engineering methodologies), evaluation is an important aspect to improve the final quality of the product. Since SAW is the goal, if there is no SAW evaluation, it is hard to actually judge the usefulness of the project regarding SAW support. Others form of evaluation are also a part of the indicators, although not as important as a SAW evaluation.

Another indicator is adaptive UI (personalization) considering user expertise. Since different workers, regarding level of expertise, have different process to acquire SAW, customization, at least considering expertise, is an indicator of SAW support.

The next indicator is the level of involvement of the user in the SA computational process, as detailed by Botega [Botega16]. There are three levels of involvement: 1) passive, where users are only a consumer of information; 2) active, where users interact only with the

final result of the SA processing; 3) pro-active, where users can influence and interact during the SA processing.

Finally, a set of 8 indicators represent the 8 basic guidelines from SAOD, explained in Section 2.3.2.2.

Many other indicators could be used, like the use or not of semantic reasoning (ontology), machine learning capabilities and others. Nevertheless, the indicators adopted were the ones more explored in the literature, and these features (ontology, machine learning), whenever present, will definitively assist obtaining some indicators, like for instance support for level 2 SAW.

Four works were chosen to be analyzed and compared to this thesis. Before this analysis is presented, it should be noted that, since most papers do not directly refer to the proposed indicators, a careful analysis of the writing, and for the UIs, of the figures, was executed to evaluate each paper. Table 3.1 shows the set of quality indicator for these four projects and this thesis. Section 7.2.2 delves deeper in the list of features of this thesis showed in Table 3.1.

Table 3.1: Quality indicator for SAW support among many works in the literature and this thesis.

Criteria ↓	Works →	Nwiabu12	Villaren12	Feng09	Baader09	Thesis
CTA		Scenarios-based design	Situation Model	-	-	GDTA, FSA
SAW eval		-	-	-	-	SART
Other eval		Usability Measurement	-	Performance	-	Performance
Personalization		-	-	Yes (entity agent)	-	Yes (Personal agent)
User involvement		Active	Passive	Active	Active	Active
SAW Guidelines	Goal-driven processing	Situation-based Task Model (HTA, scenarios, requirements)	HTA Situation model	Situation model Rule-based action recommendation	User querying	GDTA Goal-based filtering Decision-driven design
	Level 2 SAW support	-	-	Level 2 SA model	Level 2 SA model	GDTA Pre-made UI components (visual juxtaposition) Level 2 SA model
	Level 3 SAW support	-	-	Hardcoded Level 3 functions	Level 3 SA model (alarm system)	GDTA Pre-made UI components (visual juxtaposition) Level 3 SA model
	Global SAW support	-	-	Alarms "Big picture" display	Alarms	FSA, Agents, Alarms Partition dynamism "Big picture" display
	Saliency mechanism	-	Task transition model Adaptation model	Level 3 alarms	Level 3 alarms	Messaging mechanism Level 3 alarms
	Parallel processing	-	-	-	CNL Interface	Partition dynamism
	Information filtering	Context based UIs	Task transition analysis	-	User querying (control)	Goal-based filtering Level 2 and 3 filtering Messaging mechanism Layer of information Customization
	Rampant futurism	-	Task transition analysis	-	CNL Interface User querying	GDTA Customization Partition dynamism

Now each work will be analyzed in chronological order. This analysis will not only explain in more depth each work, but also compare their main features and quality indicators for SAW support results with this thesis.

3.5.1 A Novel Architecture for Situation Awareness Systems [Baader09]

Baader et al. developed an architecture (Situation Awareness by Inference and Logic (SAIL)) divided in three tiers, each one corresponding to a SAW level. The first tier, data aggregation, is responsible for identifying entities and their properties. The second tier, semantic analysis, evaluates entities using defined knowledge to understand relations and meaning and form a situation. The third tier, alarm generation, analyses possible evolution over time of each situation and send alarms to the operator when a possible critical situation is recognized to arise.

All three layers are specified using declarative logic to allow customization for each application (favoring reusability of this architecture).

A difference when comparing [Baader09] with this thesis, in the modeling sense, is their description logic model is more oriented to a logic paradigm (although the data layer uses predicate logic), and their description logic generates a result that can be used to further infer information. For instance, the following case generates as a result the assert “Y is the target of the agent”:

```
(firerule (and (?EM move) (?EM ?Ag has_theme) (?Ag fighter)
(?Ag ?Org associated_with) (?Org s_blueland enemy_organization)
(?EM ?Y has_direction) (?Y s_blueland associated_with))
((related (new-ind aggr ?Ag ?Y) ?Y has_target)))
```

For this thesis, the knowledge model has a more intuitive language, and is also more focused on determining states for each entity (agent), such as the following example of transition between states:

```
Ent.Legislation.NR15.Noise.PermanencyTime(Env.current) < ( Env.Current.TimeInsideArea() +
Env.Current.EstimatedTimeToExit() + Proc.InterruptionTimer + Max(Procedure.DurationRemaining,
Enterprise.Shift.RemainingTime) )
```

However, since no assert is generated in our SA model, this rule can only be used “locally” (by the agent responsible). The result of this rule, changing the agent state, can be explored by other agents (agents can ask each other their states). A future implementation of our solution could consider the description logic from [Baader09], as long as the impact on UI

design and generation are fully considered (thus, a research to explore this solution is required).

Regarding level 3 of SAW, both works present similar solutions. [Baader09] uses a Buchi automaton as a DFA to generate alerts when due, and in our solution a similar concept is explored. However, we go beyond only generating alarms, allowing the level 3 computations to become any necessary change to the UI (since the designer is the one to determine how the UI will respond to the situation).

Furthermore, although the description logic may be used in powerful ways to model level 3 SAW, their specific proposed solution may have “wasted” the potential of automated reasoning, because their description logic for this case (a linear time temporal logic - LTL) does not generate new asserts (new information to be reused by the inference machine). Thus altogether, they are actually only using a complicated Finite State Machine (but not more powerful than ours) and only to generate alarms.

When comparing SAW quality indicators between works, the main differences arrives from a lack of cognitive analysis for their UI design processes, lack of any evaluation, and no support for Global SAW beyond the alarms.

An advantage of [Baader09] is their Control Natural Language interface that allows for easy user querying for specific information, guaranteeing the guidelines of parallel processing (by the usage of different medias and modes – sound), information filtering (since user can request the information only when they desire) and rampant featurism (since new features do not imply new information, because users are responsible for requesting information).

3.5.2 Modelling situation awareness for Context-aware Decision Support [Feng09]

Feng et al. built the system CaDS (Context-aware Decision Support), that uses a SA model together with entity agents to generate a personalized view and service for users.

Each user has their own entity agent, responsible to communicate with the SA model and adapt (adaptive) UIs based on the context. Agents are also able to perform event classification and action recommendation in a proactive manner.

Their implementation of the entity agents is based on the production rule representation and a logical reasoning mechanism. As analyzed in section 3.3.1, the use of

productions rules is also approached in this thesis, but to create DFAs instead of feeding an inference engine.

In the end, their agents is quite similar to this thesis proposal, with the difference that they create only one agent to each user (making the agent really big), while in this thesis, SA inferring is distributed among several agents (since agents are modeled based on any type of entity, and not only user). This approach was adopted to facilitate for SMEs to create agents incrementally and iteratively, instead of having to create a “big” agent at once (for each user).

Their biggest disadvantage is the lack of support for any form of model-based or declarative based modeling. Their system is hardcoded, regarding SA model, limiting its capabilities to functions listed in their paper.

Regarding SAW quality indicators, besides not having a CTA and SAW evaluation, their biggest problem is a lack of an extendable level 3 mechanism. They also lack considerations for parallel processing, information filtering and rampant featurism.

3.5.3 Modeling Task Transitions to Help Designing for Better Situation Awareness [Villaren12]

Villaren et al. proposed a methodology to define an enhanced task transition, regarding UIs, to preserve user SAW. Their focus is on changes in the situation, particularly those caused by tasks transitions, and how this may affect the user SAW and how to mitigate any possible harm to disorientation and loss of SAW in these cases.

Their methodology have five steps: 1) define tasks in a Concur Task Tree (CTT); 2) define SAW using DSA theory; 3) associate task and SAW models; 4) define and classify transitions; 5) recommend UIs to deal with transitions.

Step 4 is executed through the Tversky’s similarity model, which measures similarity of 2 items (2 nodes of the DSA graph), followed by a graph matching algorithm (coupling, pairing or a set of independent edges) and salience extraction (vertices that have degree $-d(v)$ -equal or higher then 5, in other words, nodes that have 5 or more connections, are considered salient).

The usage of metadata to the DSA graph is also possible, to facilitate posterior UI designs. Metadata such as importance, or preferred visual presentation format (number, graph) are associated with nodes, but this process is manual.

In the end [Villaren12] presented possible UI changes for the game Battlefield 3™, with most of them resuming as an additional animation to change cameras between first and

third person (to assist in user orientation during this process). They promise more recommendations and to automate tools to execute their methodology in the future.

Comparing to this thesis, many similarities arises. Their use of DSA, hierarchical task analysis, model-based design, the combination of task and SAW models and the considerations of impacts in HCI based on this combination, are all similar points in both works. Even their transposal of DSA theory from a multi-operator / single-task paradigm into a single-operator / multi-task paradigm was also adopted for this thesis.

Nevertheless, [Villaren12] focus is solely on improving task transition UIs, thus not offering solutions to design the “normal UI” (the UI used whenever a task is going on).

Considering SAW quality indicators, [Villaren12] was hard to analyze because the proposed UI is actually an increment to a current game (Battlefield 3™) UI and they also only proposed UIs for transition between tasks. Therefore, most SAW guidelines were not achieved. Another important flaw is their disregard for SAW levels 2 and 3, since their work is solely based on DSA theory, and this model does not considers Endsley SAW model.

Their advantages is that their task transition analysis allow to implement the guidelines of salience mechanism, information filtering and rampant featurism, by using an adaptation model to highlight important information during tasks transitions.

3.5.4 User interface design for situation-aware decision support systems [Nwiabu12]

Nwiabu et al. describes in their project a design process for “Situation-aware User Interface”, using declarative models and MBD. Therefore this is theoretically the closest project to this thesis in similarity. However, as is explained below, the term Situation-aware was used “lightly” by the authors, as, by the analysis to be provided, Context-aware is more fit. In fact one of the reasons this project was chosen to be explored in depth is the necessity for this thesis to elucidate and clarify on the differences between context-aware, situation-aware and Situation Awareness support. This topic was already briefly discussed in Chapter 1, but now it will be discussed with examples from [Nwiabu12].

So, what is a situation and what is a context? In the context-awareness community, context is classified as anything that is important, and sometimes the term situation is used as a higher level context (situation-aware). In the SAW community, situations and context have the same meaning (although the word context is not usually used). The difference is that context-aware community studies the adaptation of systems, while SAW community studies

the capabilities of system to support the user decision-making process (in a complex dynamical environment).

A SASS (Situation Awareness Support System) may or may not be context-aware, but it needs to have a comprehension of cognitive demands of the user, how goals are achieved through decision-making, and it needs to support these demands and this process (decision-making). A context-aware system focus on adapting/revealing information and functionalities (services) for the user according to context. Both can coexist and complement each other, but it is also possible for a system to have only one of these features.

[Nwiabu12] proposes a framework for UI design that allow context to be considered in task specification. In a somewhat similar goal to [Villaren12], by using this framework, transitions between situations can be defined by the designer. However, the goal is not on UI adaptations between situations, but on defining UIs for different situations. They assume three different situations (normal, warning, and danger) and allow designers to create UI models for each situation for each task.

This feature of allowing UIs to be defined for different situations was also used in this thesis, especially considering dangerous abnormal situations.

Nevertheless, [Nwiabu12] presents several problems (which may have been clarified through a more thoughtful writing or an extended paper). Their major problem is that they transformed SAW into a task with a specific time to execute. But others problems creeps out, such as a lot of talk about changes based on context, but nowhere in the paper they explain when (or where) contexts are defined and how they are defined by designer (and even **if** they are defined by designers or not).

Finally, as clear by their SAW quality indicators, [Nwiabu12] seems to be more focused on context-awareness rather than SAW, as shown in the following quote from them:

“Situation-aware interface design is an SA-Oriented Design (SAOD) that adapt to the current situation with emphasis on the whole system.”

[Nwiabu12] also lacks support for levels 2 and 3 SAW, and the guidelines of Global SAW, salience mechanism, parallel processing and rampant featurism. Their advantages are they offer goal-driven processing because of their situation based task model, and information filtering due to their context-based UI.

3.6 Final Considerations

As User Interfaces for industry are further developed and enhanced, requirements to improve weak spots begin to surface. One of these requirements is to improve user Situation Awareness during tasks, which can lead to improved decision-making, reducing errors and improving efficacy and efficiency.

To create UIs with SAW support for maintenance, initially this thesis reviewed the current state of the art of maintenance UIs and Situation Awareness Support System and UI development.

From this analysis, a solution was devised involving interface agents, multiagents architecture, and a model of Situation Assessment.

To implement this solution, the model-based approach was adopted, especially for its favoring and allowance of an End-User Development paradigm, which is crucial for industry and maintenance due to its unpredictability and constant evolvement of its processes (and changes in the equipment).

Therefore, in this chapter, all the important solutions influencing and molding this thesis were analyzed. The chapter closes with a discussion and comparison of current solutions to SASUI design. The discussion is for SASUI for generic domains, since no one, as far as the author knows, explored a SASUI design process or even a single SASUI for industry.

In the next chapter, two important contributions of this thesis are presented: first, the Framework of Situation Awareness Aspects, that allow for a structuring of SAW to have a holistic vision of the situation when designing UIs; second, the autonomous SAW-driven interface agents and the MAS architecture designed to support it, allowing for easy design by SMEs of new UIs while having a powerful model of SA for run-time generation and intelligence of the UI.

Chapter 4

A MULTIAGENT ARCHITECTURE FOR USER INTERFACES THAT SUPPORTS SAW

4.1 Initial Considerations

This chapter will present a multiagent architecture that both instantiates, at run-time, agents (Autonomous SAW-driven interface agents) defined in the design process, and allows the presentation of information (in the UI) from many systems in the industry (e.g. E-Maintenance).

To better understand the underlying features of the architecture, the introduction of the Framework of Situation Aspects (FSA) and of the autonomous SAW-driven interface agents is done beforehand in this chapter, followed by the presentation of the Architecture.

The Framework of Situation Aspects (FSA) is a conceptual framework that maps all the important information that constitutes the user's SAW in any domain. It clarifies key factors, variables, and the relationship among them.

The Autonomous SAW-driven interface agents are computational and UI agents that form the basis of the UI modeling process proposed in this thesis. They are modeled by the SME based on the SAW requirements of the user, each one providing support for important decisions the user will make.

4.2 Framework for Situation Awareness Aspects

The Framework of Situation Awareness Aspects (FSA) is a conceptual framework that provides a state oriented view of SAW (as opposed to a process view), by defining the possible knowledge (state) of a situation users may possess. The FSA structure heterogeneous sources of information in an organized knowledge representation model.

It is visually represented by a heterarchical-hierarchical graph that expresses “a link between situation and an internal representation of the elements present in the situation” [Bossé07], distributed among 7 aspects (or awarenesses) and influenced by many factors and variables.

It can also be considered similar to a knowledge network or mental model or even a propositional network from DSA. It defines “what is a situation”.

FSA has two secondary benefits: 1) it assists complementing others cognitive tasks analysis (such as the GTDA); 2) it represents an initial view of the data structure desired for the UI, and can be translated to such with the assistance of tools.

However, it should be designed based on the desired User Experience, and therefore has a focus on providing and using information to adapt and display in the UI (and not only mapping an expert’s mental model).

An FSA should be defined based on a holistic study of the desired domain processes. To create one, we initially recommend mapping the different types of information essential to the user and then categorizing it into different aspects. Each of these aspects represents its own “awareness”, based on a set of information that should ensure the best service-level possible. To respond to a broad range of contexts, these aspects should be combined to generate UIs that support user Situation Awareness development.

The goal of FSA is to better define SAW in a particular domain, providing an up-to-date framework that can be used to identify SAW problems and point potential solutions related to each aspect. The framework will help understanding SAW in a domain and assist UI designers know at what time information is useful to users. It can assist designers in leveraging users’ adherence by providing a dependable source of information to optimize display according to the situation.

For this thesis, the FSA was used to define a knowledge representation model for maintenance. Future developers and designers can use it for their own application/domain.

To establish the set of aspects that composes FSA for the study case of this thesis, the initial step was mapping the information typically required by maintenance technicians to ensure an effective maintenance in the field. This material was used to understand the maintenance activity and the main inputs for the decision-making process.

Figure 4.1 shows the main awarenesses (aspects) that compose this framework for industry. Each awareness is a set of information or concepts that must be cognizant to the technician during a task, although different tasks demand a different composition of these awarenesses. Awarenesses also have information that can be used by the UI to adapt itself.

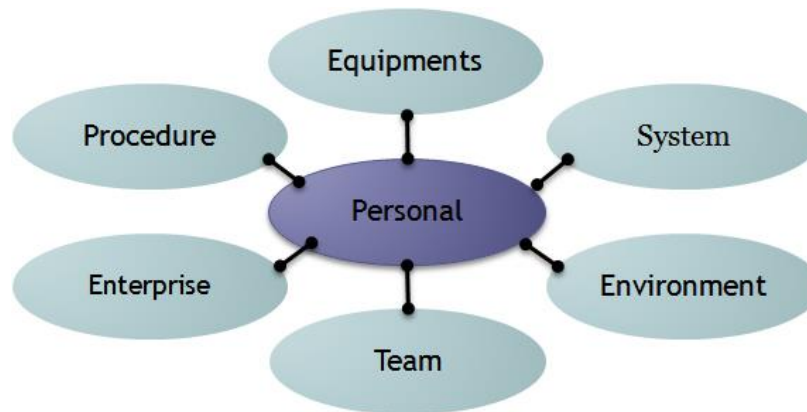


Figure 4.1: Framework of Situation Awareness Aspects (FSA) for maintenance and its 7 awarenesses or aspects.

The flowchart was generated according to interviews with experts and analysis of the works of [Dhillon06, Endsley00a, Chinoy11, Lynas11, Sneddon13, Golightly13, Bullemer13, Carroll03, Nazir12], among others.

The initial FSA structure was then proofread by specialists. The final result, with all the aspects of a maintenance situation are presented below, in addition to an analysis of impacts of the 7 aspects on the UI and possible UI guidelines, followed by an identification of some common problems of SAW in each.

4.2.1 Personal Awareness

Personal Awareness is related to the user and his/her characteristics and should be taken into account to create UIs that are compatibles to each user mental model. Factors such as ability (experience, training and skill), general cognitive level, motivation and stress should be considered to interfaces adaptations.

Personal Awareness, **Figure 4.2**, is mostly useful for guiding visualization adaptations in the UI.

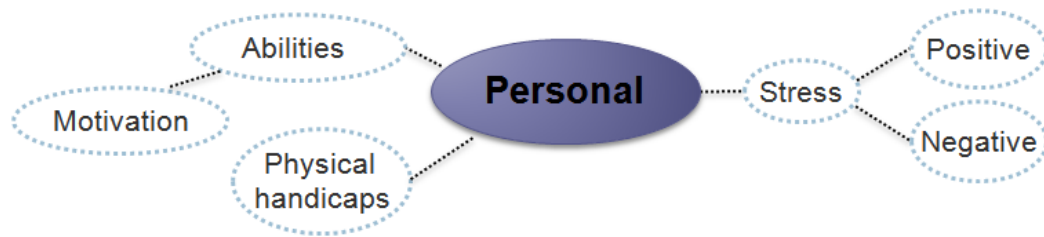


Figure 4.2: Personal Awareness in maintenance.

'Abilities' and 'motivation' are two important factors responsible for errors in industry operations. Their relation is crucial because although higher ability indicates possible higher productivity, motivation is the leverage that controls the final outcome of efficiency. Although motivation is not easily measurable, there are techniques that can help motivate users, such as Gamification [Deterdin08].

'Stress' can be divided in positive and negative. Yerkes-Dodson's law postulates a relationship between stress (named as arousal) and performance [Yerskes1908]. However different types of task (skill, rule and knowledge based) react differently to stress [Yerskes1908]. Methods do exist for the user to tell his/her overall stress level, like the Personal Kanban [Benson11], however this information remains difficult to accurately measure.

Finally, 'physical handicaps' are important Filters to be considered. For instance, deafness can change the modes of output of the UI to not use sound.

A simple method to use these and others personal factors is to create profiles (similar to Personas) and adapt UIs to them, instead of each factor. Thus, a stressed experienced user could be fitted in the same profile of a novice user.

Some of the common problems to SAW related to Personal Awareness are:

- Lack of experience, preparation and training in the work and evaluating risks [Endsley12, Chinoy11, Sneddon13];
- Lack of motivation [Dhillon06];
- Long work shifts with no rest [Sneddon13].

4.2.2 Procedure Awareness

Procedure Awareness is the understanding of the procedure and comprehension of how does it have to be executed and how is the execution going on. This awareness is the one that most benefit from experience, but also one that can be increased by complexity and occupies a great deal of the working memory. Having Procedure Awareness, means knowing

what needs to be executed in the work routine of specific equipment and the state of the current procedure.

One of the data requirements for obtaining Procedure Awareness are the maintenance 'actions' required to perform the procedure. 'Actions' are broken down into supporting components. 'Common mistakes' are the re-occurring issues the user needs to be aware of. 'Interruptibility' indicates if the flow of the procedure can be interrupted or not. 'Type' denotes actions that are irreversible or simple. 'Decomposition' is the subdivision of an action in a verb (unscrew), tool (screw driver) and piece (screw).

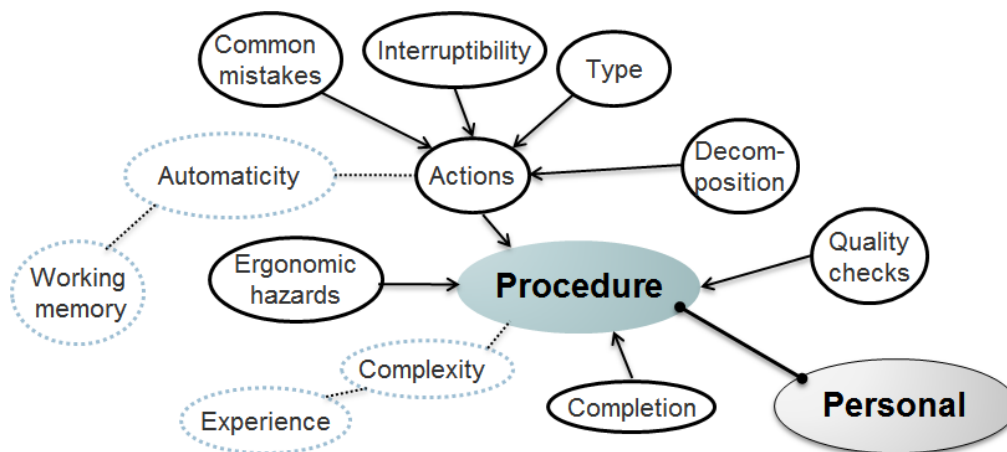


Figure 4.3: Procedure Awareness in maintenance.

'Actions' can be influenced by 'automaticity' and 'working memory'. As the user executes a procedure more times, actions become more automated, instructions of actions are memorized and thus less working memory needs to be used to have Procedure Awareness [Endsley12]. Experience helps develop automaticity, but, according to Endsley and Jones [Endsley12], human working memory is a big limitation in SAW and systems need to consider it in their design, or else errors are possible to occur.

This Awareness is also dependent on the 'complexity' of the procedure and the 'experience' of the user [Harris08]. While they may appear similar to the 'automaticity/working memory' duo, the former is more about the inherent difficulty of the procedure, while the latter is more about the repetition of actions (independent of difficulty). This means that 'automaticity' can be used to change the amount of information about an action the user needs to visualize, while 'complexity' can be used to change how the information is going to be visualized (to either compensate the user lack of experience or the high complexity of a procedure).

Also required is information on 'ergonomic hazards', since some procedures need to be done during a long period of time in a posture that may harm users (identified through

ergonomic assessment techniques, such as PLIBEL [Kemmlert06]). Also composing Procedure Awareness are 'Quality checks', which are done in a few key steps to assure the quality of the work. 'Completion' is considered as users get absorbed in their work when it's approaching the end, leaving them with high Procedure Awareness but lower SA (as described in an accident in [Golightly13]).

A solution to maintain Procedure Awareness is using Interactive Electronic Technical Manuals (instructions) augmented by Augmented Reality, so that it is possible to view projected in the equipment instructions and Computer-Aided Design (CAD)/Computer-Aided Engineering (CAE) data [Espíndola13].

- Some of the common problems to SAW related to Procedure Awareness are:
- The task tends to take all the focus from situation assessment [Golightly13];
- In the end of a task, the focus is even more reduced in situation assessment [Golightly13];
- Repetitive task results in complacency [Sneddon13];
- Not understanding the work/task proposed [Golightly13];
- Distraction/lack of focus during work [Golightly13].

4.2.3 Equipment Awareness

Equipment Awareness is the understanding of the equipment and its behavior. It is generally considered the most important Awareness in maintenance, even defined as the concept of SAW in maintenance [Endsley00a] and is particularly useful to assist in diagnosis of problems.

Equipment Awareness, **Figure 4.4**, represents the understanding of equipment and its behaviors and risks. The main node is the 'target equipment' and from this, information to support 'diagnosis' and to avoid 'risks' is priority.

Diagnosis can be supported by the specialization 'troubleshooting' [Harris08], which has several strategies, from traditional, to structured troubleshooting and negative reasoning. To support it, information coming in real time (or records) from 'sensors' can help evaluate the source of the problem. There is also 'design' information (CAD), which can provide an understanding of the equipment functionality, and 'age cues', which are useful for identifying wear (leaks, fatigues or missing pieces) by comparing virtual to actual age. Finally, 'Records' can assist analyzing the problem, with 'Reliability' of specific components and malfunctions expectations also aiding the diagnosis process [Harris08].

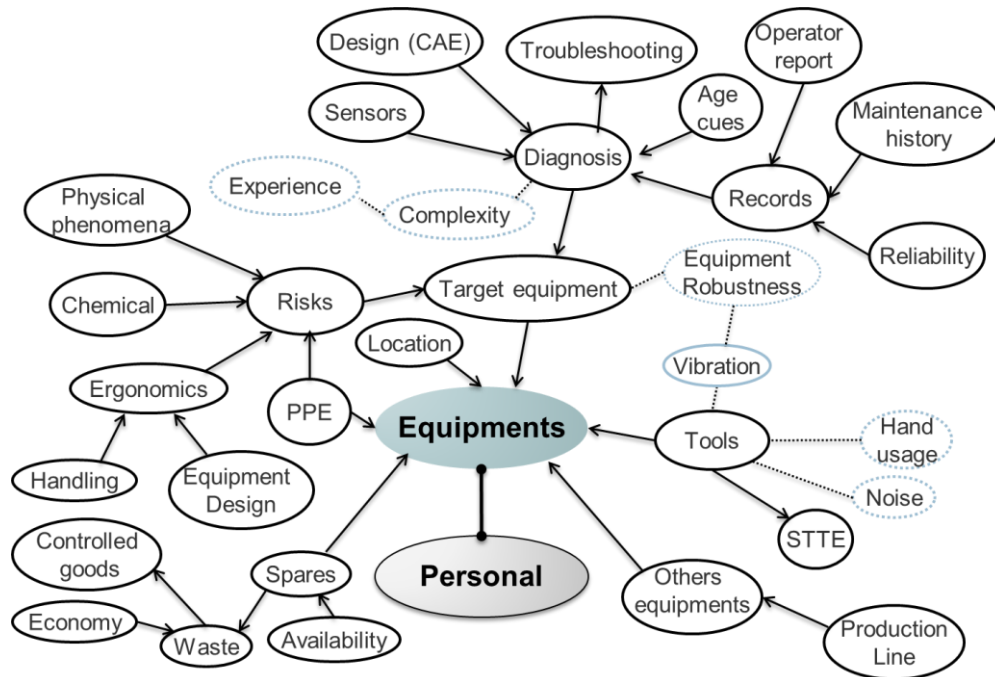


Figure 4.4: Equipment Awareness in maintenance.

Furthermore, 'complexity' is useful to determine the information density for the UI in the diagnosis processes. In this case, 'complexity' relates solely to the equipment and is proven to be balanced by 'experience' (and system design and motivation) [Harris08].

'Risks' are used to improve user safety during a task. Each equipment has a different set of risks while conducting repairs, and they are normally 'physical phenomena' (heat, cold), 'chemical' (fumes, radiation, acid) or 'ergonomic'. An example of ergonomic risk is lifting ('handling') equipment with sharp edges ('equipment design'). The common tools to avoid risks are 'Personal Protective Equipments' (PPE), together with adherence to safety regulations and procedures (see Enterprise Awareness, subsection 4.3.7).

'Tools' is another element in Equipment Awareness, some of which could be the specialization 'STTEs' (Special Tools and Test Equipment). Some tools require both 'hand usage' and this information can be used if designing the maintenance UI for a Smartphone. Noise from a tool can be used to adapt a UI to not use sound cues (the same outcome of the 'physical handicap' deafness) and can also influence 'stress' level [Helander06]. Finally 'vibration' is contrasted by the 'equipment robustness', because fragile equipment may demand extra care in using some tools. In addition, vibration of 10-25 Hz decreases the visual acuity [Helander06], thus the output media could be adapted in this circumstances.

Another Data node is 'Other equipment' that can affect the 'target equipment' or the user. For instance in a Power Plant, it may be necessary to electrically isolate other equipment to be able to work on the target equipment.

'Spares ' are the availability of the replacement part and the status of the broken pieces. Removed parts are 'waste', however the handling of these parts are governed by 'economy' concerns focused on having the best return by minimizing loss, possibly by trying to rebuild or sell the item. This is additionally influenced by 'controlled goods', specialized items that must be handled in a way that may not be in line with the best economic policy (often due to technological control, military, radiation or legal/statute requirements).

Regarding UIs guidelines, two major aspects of this awareness are risks and reliability. Risks should ideally be treated as an event with: association, duration, area-of-effect and priority.

- Association: useful to determine the entity responsible for this risk, so that an UI for a risk can be designed once and used for every procedure related to the entity;
- Duration: important for the process of maintaining awareness during a task;
- Area-of-effect: specified so that coworkers can be aware of the range of a risk;
- Severity and likelihood (priority): mostly used to avoid information overload, since higher priority risks should have better salience (such as the color red or a flashing light).

Whenever possible, Augmented Reality is the best media to display risks, because it can lessen the cognitive processing by displaying a risk on top of its source. Nevertheless, studies on how to project risks using AR are currently lacking.

Reliability is how close the equipment is to a breakdown, verified through age (Reliability-Centered Maintenance), prognostics and reading of sensors (Condition-Based Maintenance). If the status of the equipment being worked points to an imminent or close breakdown, UIs should inform users and display risks associated with it.

To keep Equipment Awareness, besides using risks and reliability, UIs should allow users to view on demand information about equipment and they should focus on assisting diagnosis of problems, with a list of general problems and their visual cues.

Some of the common problems to SAW related to Equipment Awareness are:

- Confusing translation of documentation of imported equipment;
- Specialists failing to understand the functional relationships of equipment when problem solving [Chinoy11], normally by use of incorrect mental model [Sneddon13];
- No documentation update [Chinoy11, Golightly13];
- Equipment too complex and risks not known to user.

4.2.4 System Awareness

System Awareness in maintenance is being aware of the computational, electrical, mechanical and hydraulic systems and how they control the process, in simpler terms, the automation in the system. Automation is the smart management of a system using technology, so that it operates without human direct control [Lynas11].

System Awareness, **Figure 4.5**, is knowing what the system can do by itself, and how can the user control it. This awareness represents automation in industry, in which the human element is still important [Endsley12, Harris08] and therefore a human-centered system must provide information necessary to cope with the level of automation in the workplace.

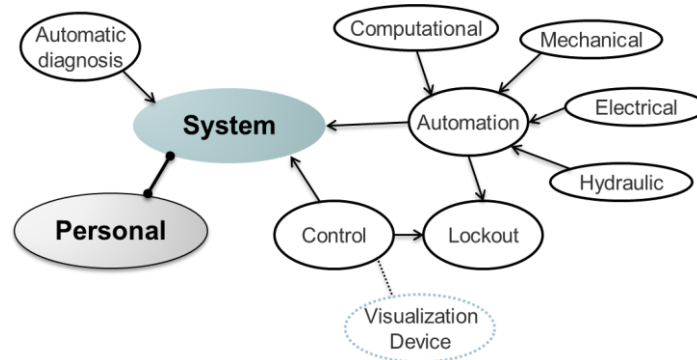


Figure 4.5: System Awareness in maintenance.

In general, 'automation' is 'computational', 'mechanical', 'electrical' or 'hydraulic'. In the FSA, the 'automation' Data node is focused on how the industrial plant and some of its equipment react to the user (and vice-versa) during maintenance activities [Lynas11]. To maintain this awareness, users must be informed of the automated functions of the system that can affect their work, such as an air compressor that starts automatically when the compressed air supply is low.

Automated systems allow users some level of 'control'. In maintenance, this control is exercised mainly with 'lockouts' of equipment. For instance, a lockout would prevent the air compressor to turn on during its repair.

It is contended that totally 'automatic diagnosis' is impossible [Harris08], and therefore a mid-level automation with human-centered design is the right approach. Thus, the 'automatic diagnosis' node must provide data for users that complement the automation solution designed. For instance, this solution could identify probabilities of problems and show the logic behind these probabilities, so that users may understand the system and take the final decision.

Finally, another particular aspect of System Awareness is the 'visualization device'. Comprehension and ability to use this device can influence a user Situation Awareness, thus users need to be trained in the device before engaging any activity. The 'visualization device' can be used to determine some limitations in design, such as the screen size limiting information density.

There is a high potential in automations to distract, overcharge and confuse operators, instead of supporting them. Therefore this awareness is centered on users understanding systems controlled actions and how they influence his/her work.

With the growth of automation, a new set of human factors have arisen, such as: how is information displayed in automated systems; how is the system controlled; do the operators accept the system; what happens when the system fails; boredom associated to a change in work from execution to monitoring; how does the system deal with SAW; disqualification and behavioral changes in operators depending on the level of automation and how this impact in risks.

Several levels of automation can be defined to increase or decrease automation in a system. Parasuraman et al. [Parasuraman00] defined a four-stage model for levels of automation: i) information acquisition; ii) information analysis; iii) decision and action selection; iv) action implementation modes. Examples of use of these levels include Bashiri and Mann, which created an UI to support each level and evaluated users' SAW to compare the advantages of the levels [Bashiri14].

UIs should focus on keeping user aware of automation by informing actions of the systems, as if the system was a coworker, and supporting direct system control.

Some of the common problems to SAW related to system awareness are:

- User “out-of-the-loop” with automations [Endsley12];

4.2.5 Environment Awareness

Environment Awareness is being aware of the surrounding considering the risks and consequences of the work and it is essential for keeping a high level of SAW. This awareness is considered by Nazir et al. as the Situation Awareness in industrial field work [Nazir12] and defined as the perception of changes in the physical parameters, the comprehension of the changes and anticipation of consequences.

Regarding FSA, 'Environment' Awareness, **Figure 4.6**, is about knowing risks, the terrain around the user and sustainable development strategies to preserve the ecosystem.

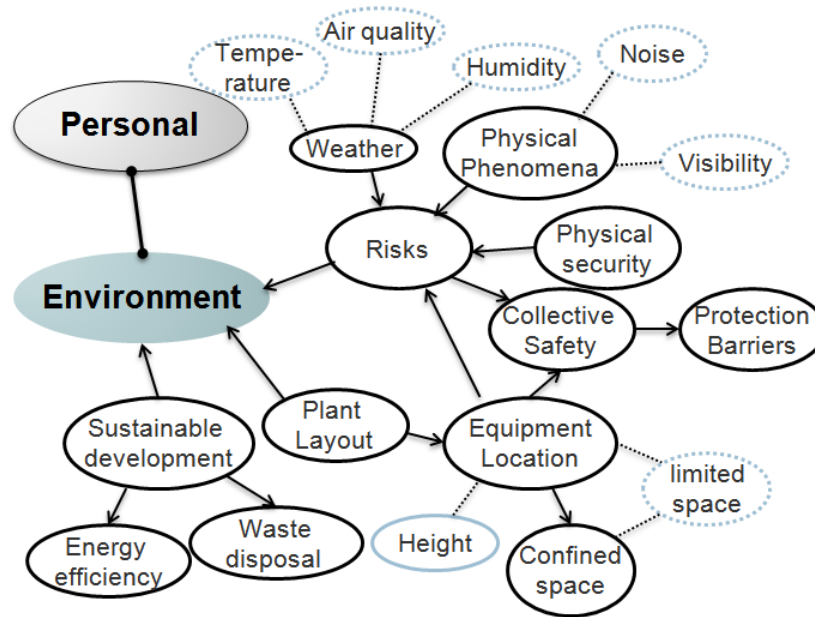


Figure 4.6: Environment Awareness in maintenance.

The key Data node is 'risks', and they are classified as 'physical phenomena' or 'weather' based. 'Noise' and 'visibility' are examples of the former and 'Temperature', 'air quality' and 'humidity' of the later.

In general, 'physical phenomena' can be used to adapt the UI mode for input and output. In addition, 'noise' has been proved to affect knowledge based tasks (in which the user needs deep thinking) such as diagnosis [Helander06].

Harsh weathers may reduce physical and psychological functions. Heat may cause sweat (affecting grip), distractions and psychological strain [Parsons03]. Cold can induce a loss of sensitivity and stiffness of fingers, overall discomfort and shivering (causing distractions) and may induce arousal and stress [Parsons03].

Another common risk is based on the 'equipment location' at the 'plant layout'. To prevent risks from the equipment to other workers, 'collective safety' has to be provided by using 'protection barriers'. Two examples of barriers are Insulating Protective Equipment to cover equipment (blanket) and barriers in train tracks. These barriers help assure safety for everyone involved or near the activity.

'Equipment location' is also a risk, some equipment are mounted at height and need ladders or scaffolds for access, thus inducing a fall risk. Other equipment may be in 'limited spaces', or even a 'confined space' (that has a specific work legislation to execute tasks at).

'Sustainable development', posits an economic development without depleting the environment. 'Energy efficiency' (with techniques such as eco driving, for driving while

saving gas) and ecological 'waste disposal' are two of the strategies that can help achieve a better balance between industry and natural environment.

In some domains, barriers are an important tool for safety, because they can prevent dangers in the workplace. Therefore UIs should take advantage of Barriers and inform users when they are present, to assist maintaining Environment Awareness.

UIs should also be designed to with the following principles:

- Avoid risks: in an industrial ambient, dangers caused by high temperatures, electricity, radioactivity, etc, are common and human-centered interfaces should focus on keeping users aware of them when they are present;
- Adapt to visibility issues: in case of diminished visibility (fog), Augmented Reality is a good technology for path guidance, being able to display the factory layout;
- Noise limiting UIs: workplaces with heavy machinery have loud sounds or noises that make it impossible to use sound/speech based interfaces. Therefore, they must not be used in this scenario, even with the increase in processing capabilities;
- Provide sustainable development: correct and ecological procedures of waste disposal should be ensured by a human-centered interface.

Some of the common problems to SAW related to environment awareness are:

- Unexpected leakages;
- Lack of organization and cleanliness and unclear procedure of waste disposal;
- Mistakes on understanding the factory layout (access to the wrong equipment) [Nazir12];
- Lack of visibility, because of darkness (night) or weather (fog) [Golightly13, Nazir12];
- Risks not known to user.

4.2.6 Team Awareness

Team Awareness is an essential element in a maintenance process, important for coordinating and maintaining a collective awareness as a group with the same goals and because 40% of accidents in industry are caused by lack of Team Situation Awareness [Bullemer13]. As stated by Endsley and Robertson [Endsley00a], errors frequently occur in team work because information is not shared or passed between teams. Therefore, individuals need not only to comprehend the task they are doing, but also what the team is doing, because this will influence their decision-making and performance.

Team Awareness, **Figure 4.7**, has generated considerable research [Endsley00a, Golightly13, Stanton06]. Team Awareness is the understanding of the team actions to coordinate the work and the workplace with a group of people. Even a solo task can influence Team Awareness, due to a shared space.

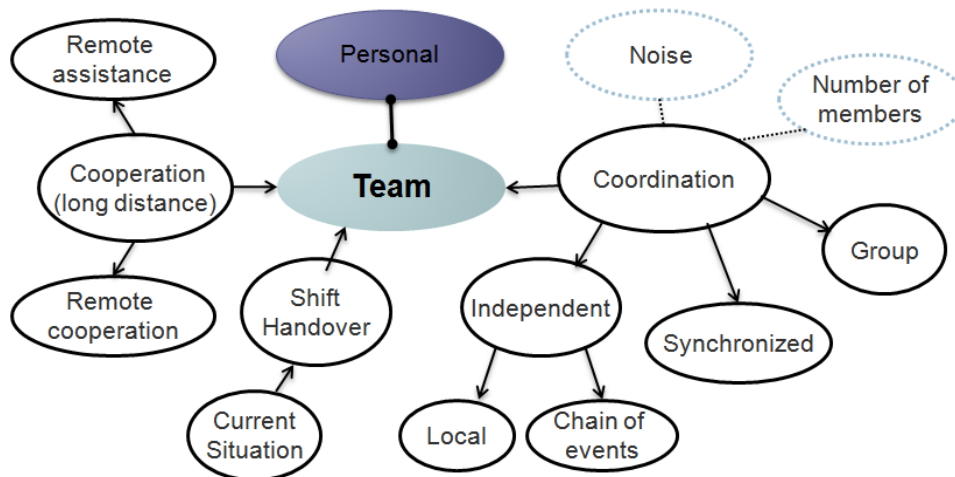


Figure 4.7: Team Awareness in maintenance.

Considering tasks with present workers, 'Coordination' is the main form of obtaining awareness. Three forms of coordination in group work exist:

- 'Independent': different goals/equipments but same physical space.
- 'Synchronized': same goal and with a necessary order of tasks.
- 'Group': same goal without any order of tasks.

Each type of coordination has a different level of demand and different request of information to maintain awareness. 'Group' work request information about each worker current function, expected finish time and actions that may interfere in others work. 'Synchronized' work request all the information of group work and also dependency of tasks. 'Independent' work is more focused on informing other workers about the user actions, especially when this action can interfere with their work, because they are happening in the same place ('local') or causing a possible 'chain of event'.

Communication is acknowledged as the main feature of team work, but 'noise' can make it impossible. However, UIs can suppress the problem by showing the desired communication output of the team for the user, thus helping maintain Team Awareness. The 'number of members' in a team can also affect awareness, since bigger teams can change interfaces to be less detailed about each team member actions.

Considering only the guidelines above, in cases of teams with 5 or more members, information overload will be inevitable in synchronized and group work. Therefore a guideline was introduced for this case. In Synchronous teams, only tasks blocking the user

should be displayed, and the current task of coworkers should be hidden, but possible to access when desired. In case of Grouped teams, only one message about coworkers should be displayed at a time (others messages could enter a queue and be displayed in order).

Team Awareness is also composed of 'shift handovers'. In these situations, accidents are common because the new worker has no knowledge of the current situation and recent occurrences [Golightly13]. Therefore, the new user needs information about changes in the overall state of the plant, such as broken equipment or events that happened, current intervention by external teams, and the work progress. Logs are generally used for this, informing events that happened during the last user turn. However, the new user should also be informed of: current progress of the task, current step to execute, current state of teammates.

Finally 'cooperation' is the last Data node for this awareness. Cooperation differs from coordination because it is team work separated by long distances (space and time). Cooperation may specialize in 'remote assistance', which is real time assistance from a distant and more experienced co-worker, that assist the user in understanding and fixing the equipment. Another form of cooperation is 'remote cooperation', which is cooperation over time, such as maintaining airplanes through different airports. In this last case, information needs are similar to a 'shift handover'.

Some of the common problems to SAW related to Team Awareness are:

- Poor communication with the team [Endsley00a], especially of critical information [Chinoy11];
- Poorly prepared planning and last minute changes [Golightly13];
- Shift handover trades a user with developed SAW for another with low initial awareness [Bullemer13];
- Number of stakeholders too high for a manageable communication [Chinoy11].

4.2.7 Enterprise Awareness

Enterprise Awareness is being aware of company controlled factors that influences the work, such as shift scheduling, workload/deadlines, assets, standards/terminologies and ethics rules of the work place. Each stakeholder interested in the user following a normative code should be represented.

Enterprise Awareness, **Figure 4.8**, is the understanding of information created or controlled by the technician's company/employer and how it can affect the work.

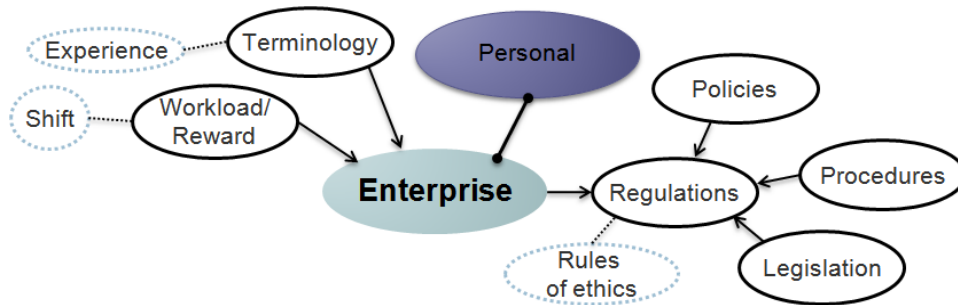


Figure 4.8: Enterprise Awareness in maintenance.

'Policies', 'legislation' and 'procedures' are the core of this awareness and compose 'regulations'. 'Policies' are useful for assisting choices in cases with multiples choices, such as “you can only use new tires”; “just a qualified technician can repair this”. 'Procedures' are for how something should be executed, for instance reporting work. 'Legislation' could be Federal laws and are rules the employees have to follow for their work, such as "the worker must not just fix an equipment, but also report, in the case of a radiation leak".

Other facets of Enterprise Awareness are 'terminologies'. They are unique to each company and new employees take time to learn. A maintenance system with voice recognition could assist inexperienced workers, for instance translating acronyms in real time.

Finally, 'workload' and 'rewards' affect the user motivation, which can change productivity, as discussed in subsection 4.3.1. Also, as discussed in section 4.3.6, the end of a work shift is a moment of low overall awareness that may facilitate accidents [Golightly13]. Therefore a UI can emphasize this lack of awareness and motivation, the former by decreasing the necessary level of Procedure Awareness in the end of the shift, and the latter by using concepts as Gamification [Deterding08], where users are shown in real time the rewards for their work (using salary and performance indicators) [Oliveira15].

Shifts, deadlines and workload can influence a person awareness ratio. Small deadlines or big workloads can cause stress, night shifts changes the circadian cycle and both can diminish cognitive capacity, impairing SAW [Kiassat13]. Also close to the end of a shift, users tend to focus only on the task and forget their surroundings; therefore UIs should further highlight events in this period (such as risks).

Additionally, because of the importance of the social component in the interface, studying the ethics and behavior rules of the work place are essential for guiding the design of a behavioral awareness interface. For instance, in some work places silence is mandatory, thus speech interfaces are not a good choice.

Finally, another cause of problems in Enterprise Awareness is the lack of use of standards and the different terminologies among stakeholders. Both can be ensured by an UI designed to standardize processes and nomenclature in a corporate environment.

Some of the common problems to SAW related to enterprise awareness are:

- Smaller attention to details and situations when close to the end of shift [Golightly13];
- Inconsistent documentation and terminology among stakeholders [Chinoy11];
- Pressure to solve problems in short periods of time [Sneddon13];
- Constant disruption of the circadian cycle due to 7days-7nights shifts schemes [Sneddon13];
- Unavailability of spare parts [Endsley00a].

4.3 Local and Global Situation Awareness

It is important, for this thesis, to define and distinct the terms Local and Global SAW. Local SAW is not a term expressively used in the literature. In other domains, such as C2 or aviation, Local and Global are either a matter of proximity (physical distance) from the user or the center of attention (or current goal).

The main use of the term Global SAW is presented by Endsley and Jones [Endsley12] and is used to provide 2 UIs guidelines: first, provide a “big picture” display (according to [Endsley12] “a high-level overview of the situation across operator goals”), being careful to not hide important secondary information; second, allow users to alternate goal-driven and data-driven processing (discussed in Section 2.3.2.2).

Supporting Global SAW is equivalent to supporting a UI that potentially enable users to have a holistic view of the situation, allowing a switch in their goals when required.

In this thesis, the structural division of the situation in aspects was used to classify what are Local and Global SAW. This classification is illustrated below for maintenance and industrial work to facilitate and even improve discussions of errors caused by lack of SAW.

Classification of Global and Local SAW for maintenance

For maintenance in industry, **Procedure Awareness** was established as being the Local SAW (for its importance), while the others aspects (**Personal, Equipment, Team, Enterprise, Environment and System**) as Global SAW (or just Global Awareness).

Since the user main goal in maintenance work is executing the procedure, it is safe to assume that their center of attention will be the Procedure Awareness, thus labeled as the Local Awareness for this domain. This finding is confirmed by literature on SAW in maintenance [Endsley00a, Nazir12, Golightly13, Chinoy11, Dhillon06, Chinoy11, Sneddon13, Golightly13, Bullemer13, Sian96, Robertson98], on-site observations and expert's interviews.

Since Procedure Awareness is interpreted as Local SAW, the other awarenesses form, consequently, the Global SAW. This means that the user main goal is the procedure, but other goals are important and should be supported, such as: being safe (Environment and Equipment Awareness), following regulations (Enterprise Awareness), working as a team (Team Awareness). These secondary goals are many times ignored by systems and their UIs, leading to an incomplete support for SAW which can lead to errors and accidents.

Another explanation of this duality (a more formal one) was already presented in Chapter 1, linking Global and Local SAW to decision-making criteria. By obtaining Local SAW (Procedure Awareness), users can effectively use procedural and technical criteria for decision-making. By obtaining Global SAW, users can also effectively use economical, legal, ethical and political criteria for decision-making. This set of complementary criteria can only be exerted by developing an awareness of equipment, team, system, environment and enterprise. The optimal decision is the one that considers all the criteria.

Wickens also discussed Global SAW [Wickens08], establishing a difference between SAW (dynamic) and routine procedure (static), and saying that “*the display features to support global SAW, necessary in the unexpected circumstances when things go wrong, will need to be substantially different from those that support routine performance in normal operations*”. This means that his vision of Global awareness also aligns with the vision presented in this thesis, which is a complement to Procedure (Local) awareness.

Nevertheless, whether to amplify criteria of decision-making, or to prevent confusion during abnormalities, by classifying aspects of the situation division as Global SAW, it is one of the aims of this thesis to call attention to these aspects in UI design for industry, to improve SAW development by users, consequently improving efficacy and safety (and in some cases, efficiency) through better decision-making.

To apply these theoretical concepts (FSA, Global and Local SAW) in a computational system, an architecture based on agents was devised and is presented in the next sections.

4.4 Autonomous SAW-driven interface agents

4.4.1 Agents Characterization

Agents in UIs for supporting decision-making are a traditional line of research, with agents focusing on delivering critical information for users. These agents are considered Autonomous Interface Agents [Lieberman97], for operating in the UI and being autonomous in relation to the user.

The agents proposed in this thesis are first and foremost UI agents. Nevertheless, while usually UI agents assist user navigating/understanding the UI, the agents proposed in this thesis are not UI assistants, but work assistants (and the generators of the UI). In other words, they do not assist users in using the UI, they are the UI that assists users during work.

This seems a big distinction but, considering the industrial domain, in fact it is not. Traditional UI agents assist users that work through the UI, but technicians work in the environment and the UI **is** the assistant (and not the main tool of work). This differs severely from how UIs used for other tasks function. For instance, in Command and Control, UIs are means to execute the work (the work is done through the UI).

Naturally, since in the maintenance work the UI is already an assistant, the UI agent paradigm fits it.

The concept of UI agents also naturally fits Lieberman guideline of time-shared user attention for agents [Lieberman97]. Traditional UI agents do not have full attention of the user, which is also the case for UIs in the industry domain, where user attention should be on the physical world and the manual operations he/she needs to execute. Therefore the UI for this domain can never assume it has full user attention and this reinforces the view of the UI composed by agents assisting the user.

Some other reasons for adopting the agent paradigm are:

- Agents can facilitate the implementation of SASS, by facilitating modeling of asynchronous dynamic environments and entities;
- Agents are also proposed in the literature of SA models [Jones11a] and used in many SAW projects (Section 3.2.2);
- Agents are intuitive to make analogies in real-life, facilitating for SME users to understand the MBDUI process. They also apply the concept of Mirror Worlds [Ricci15], serving as abstractions of real world entities;

- They represent a proven model in the literature for UI generation and adaptation for complex dynamic systems [Viano00].

Formal definitions of the characteristics of the agents used in this thesis are:

- Autonomous: independent of user and other processes;
- Interface: acts on the UI (and not purely on the background);
- Not conversational (and not anthropomorphic): there is very limited interaction of the user with the agents, since the focus of the thesis is not on multimodalities (although a future work could implement interaction modes) and users do not need to act to initiate the agents. Nevertheless, embodied conversational agents could be used in the future to increase user trust in the system;
- Heterogeneous: they differ in resources and problem solving strategies (and capability) [Wooldridge09];
- Reactive: agents have an active observation of the situation. They will use their own knowledge (modeled by SMEs) and inference of the user goal (through their goal-driven modeling) to observe the situation (elements of interest for the user) and take decisions (and act), by providing an adequate UI output to the user;
- SAW-oriented: agents have a dual role of building user SAW and providing UIs. To fulfil this role, they are designed using SA models while at the same time outputting (generating) UIs supported by these models.

According to and complementing these definitions, the proposed agent should represent a real world entity, such as an equipment, procedure, team, system, environment, enterprise or person, as defined in the FSA for the maintenance domain.

Agents are then interested in assisting the user to acquire awareness about their representing entity, so that users can form their SAW. Each agent will dispute attention with others and is mostly concerned with its own welfare, which in this case (welfare) means informing the user about their current situation (the situation of the agent). Agents use their models to observe the entity they represent and assist users by displaying important information in the UI, which will support acquisition and maintenance of SAW by the user, leading to better decision-making.

The agent environment, in this case the UI, is formally defined according to Russell and Norvig classification [Russel95], as:

- Accessible: information is assumed freely accessible, with no impediments to it. This is because the E-Maintenance system, which possesses information, is supposed to be

trustworthy. Problems with uncertainty and data confidence, in the real world, could exist, but for this thesis they are not considered;

- Non-deterministic: Agent's actions are considered non-deterministic because the current state of the UI needs to be considered. For instance, a UI with already too much information may decide an agent action (an output request) has priority below the current displayed information, and discard it. Therefore, although the agents are deterministic, their environment is not;
- Static: although the real world is dynamic, the UI is considered static, because only the agent's actions can change it (any events incoming from the real world are interpreted by agents).

The advantages of a multiagent system, according to Jennings, are presented below [Jennings99]. These advantages are not only applicable to a multiagent UI, but are also some of the differential that led to the decision of adopting agents, as opposed to other solutions (e.g., UI components or design patterns):

- Faster problem resolution through parallelization: in this case the problem being solved is that of users acquiring Situation Awareness. Dividing this problem among multiple agents allows each agent to supply some of the situational information to the user (the information judged as important at that time). This concept is clearly demonstrated in Figure 4.14, Figure 4.19 and Figure 4.21;
- Reduction in the data flow, because only partial high-level solutions are transmitted to other agents and the UI, instead of raw data to a central "point": each agent is responsible for managing its data and generating SAW-oriented information from it (levels 1, 2 and 3 of SAW), and using this information (plus their goal-driven behavior) to understand the situation and transmit only what is important for the user. Thus, a MAS distributes the load of understanding the situation to generate UIs, implementing a Cooperative Distributed Problem Solving (CDPS) technique;
- Better flexibility, because agents with different abilities are dynamically grouped to solve problems: agents are instantiated depending on the user task, and work together to build a SASUI.

4.4.2 Agents Model

This thesis proposes a model of agent that combines UI-driven agents and SAW-driven agents. Because of the analyzed characteristics, they are considered autonomous SAW-

driven interface agents, but henceforth will be only called as “agents” throughout this thesis. These are agents that will generate interfaces for users, but instead of using only task models to create UIs, they will create them based on SAW models.

Autonomous SAW-driven interface agents follow what has been proposed on SAOD for UIs: goal-driven design, support for levels 2 and 3 of SAW, careful filtering of information, support for global SAW and others. They also have an SA model to enable them in improving their intelligence, and to become truly SAW-oriented.

To follow SAOD, they have an internal structure that can only model decisions using Decision Components. Each Decision Component can access the agent data and then output a presentation. Figure 4.9 gives an overview of the agents.

Decision Components model literal decisions that users will take during their tasks, initially mapped through a GDTA (or other CTA) and then transposed to the agent during the UI design phase of the development process.

An example of a decision is “What are the possible risks you should be aware”. Thus users think about risks on their environment, equipment, automation, etc.

Agents can have more than one Decision Component (DC), but each generates their own UI, as exposed in Figure 4.3. In some cases a set of decisions (from the GDTA) are combined and represented as one Decision Component, with one UI (Example in Section 6.4.2).

An agent basic assemble cycle is demonstrated in Figure 4.4, starting from the creation of the agent and going to the definition of its Data Components and later Decision Components.

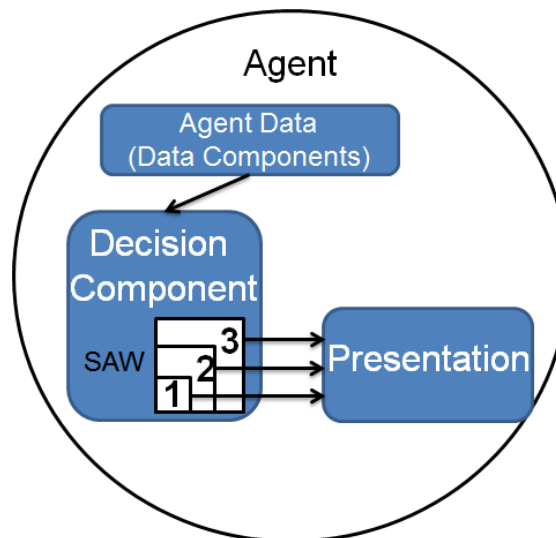


Figure 4.9: Agent overview, with data being used to model decisions, which generate presentation (UI).

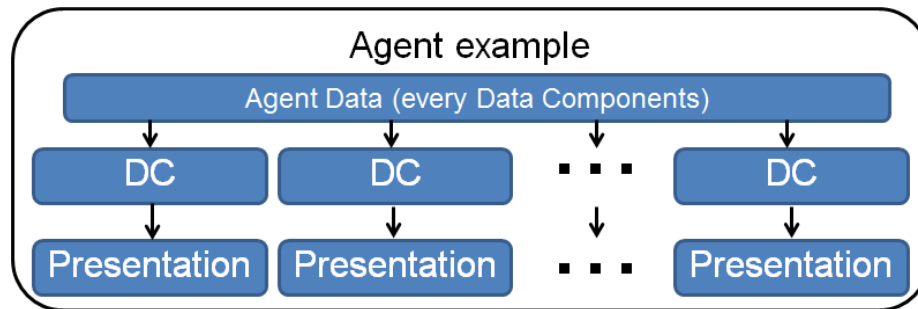


Figure 4.10: An example of agent with several decisions modeled from the same data.

Regarding agent creation, most agents are created by a group of stakeholders and implemented beforehand to be connected (mapped) to the system, so that dynamic information can be used. These are called “Existing Agents”.

Any agent that needs to use information that is dynamic should be an Existing Agent, meaning that it needs to have an interface (wrapper) implemented previously to access its data.

Information inside agents is stored in Data Components. These components are ideally mapped in the FSA, which is not only a conceptual framework to express knowledge and a CTA elicitation technique, but also the ideal data structure of the UI. Defining the agent Data Components is the action of expressing what are the data possessed by this agent.

Finally, Decision Components are defined. These components are an aggregation of: a trigger to start the decision; state machines that represent knowledge of entity situations; a UI; UI transformations; and messages passed to other agents as a form of communication. Inside each Decision Component, levels 1, 2 and 3 of SAW are modeled using the information requirements (from the GDTA). This SA model represents knowledge from the SME about possible situations of each agent. Each level of SAW is modeled in a state machine, with each state having a UI as output.

Once a decision is modeled in a component, this component can be reused by others agents to save work. A new decision may use the Specialized (SpecDC) or the Standard Decision Component (StanDC). A SpecDC is a specialization of a Decision Component to solve a specific problem. A previously modeled decision becomes a Prompt Decision Component (PromDC). These are extensively explained in Section 5.3.2.

The end goal of the combined UI (the UI generated from every agent) is to assist users in developing SAW. Therefore these Decision Components assume this goal and execute it by modeling pieces of the user SAW onto a UI. This view (of a fragmented SAW) combines with the structural division of SAW proposed in Section 2.4 and the Global vs Local SAW division discussed in Section 4.2.

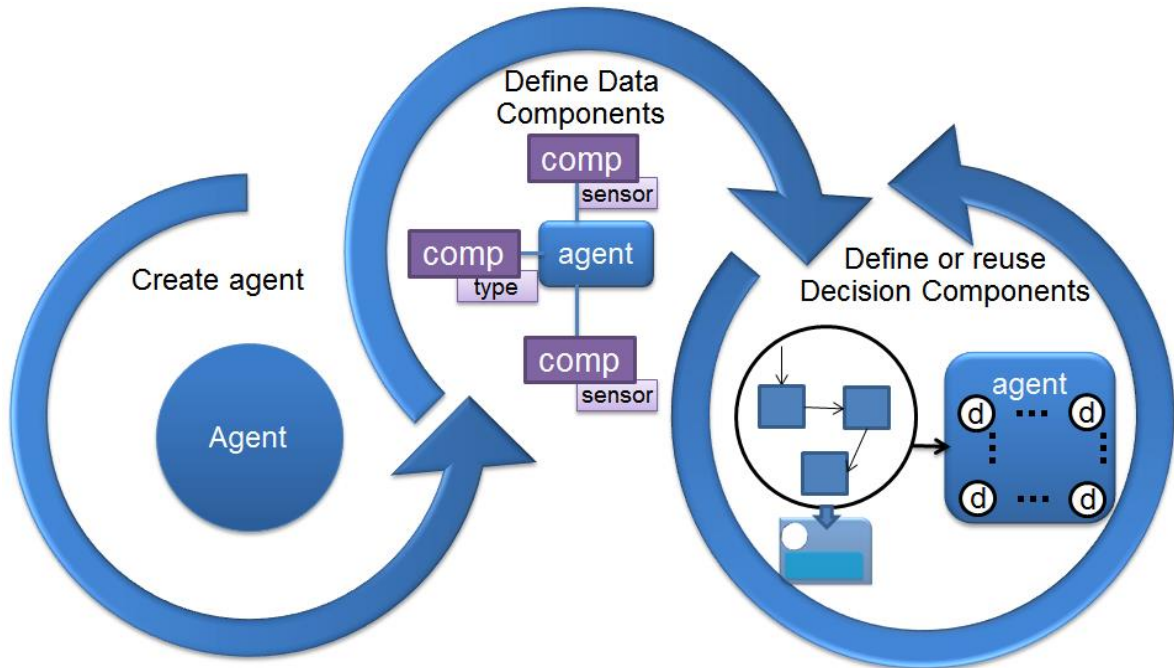


Figure 4.11: Agent assemble/life cycle, going from the agent creation, through the definition (and implementation) of the agent internal data, to the modeling of decisions.

Finally, to model knowledge (inside Decision Components) this thesis proposed a Deterministic Finite Automaton (DFA) inspired by Functional State (FS) process modelling [Halme97] to simulate comprehension and projection (levels 2 and 3 of SAW).

DFA was used to model SAW in the agents, but with some relevant characteristics: no unique transition function is defined; transitions are modeled by knowledge based rules (which resembles control policies from Functional States); has a two level hierarchy forming a structure named Decision State Graph (more in Section 5.5.3), which reflects changes in the entity situation (first level) in the UI (second level).

States and their transitions are defined by SMEs, currently based on the Production System Model [Newell72]. In other words, they are rule-based expressions of the SME knowledge. Currently transitions are defined via high-level logical equations that have access to E-Maintenance system data through the agent Data Component. Examples are presented in Chapter 5. In the future, this process can be abstracted to a visual programming solution, much like Pantagruel [Drey12].

In the next subsection, an integrated view of SAW and SAW driven agents concepts and how they are applied to generate the SAW driven interfaces is presented..

4.4.3 Conformity to the fundamentals of Situation Awareness

In practice, agents deal with SAOD guidelines in the following manner:

- Goal-driven design: agents will model decisions from the GDTA. Since these decisions are elicited from goal/subgoals, agents are in their nature goal-driven;
- Support for levels 2 and 3 of SAW: when modeling a decision, agents will have to model level 2 and level 3 SAW for this decision (as established in the GDTA). They define them by elaborating (or choosing) a strategy to combine data and translate it to higher levels. Additionally, they can directly use level 2 or 3 SAW information to display to user, to create a different UI or to adapt the UI. The set of all agents (MAS) will guide users in acquiring and maintaining SAW;
- Support global SAW: agents will be defined to cover not only Local but also Global aspects of the situation;
- Careful filtering of information: filtering is done through the goal-driven nature of the agents and by UI adaptations defined by the SME based on his/her expertise and necessities. For instance, if noise volume is lower than 90db it does not represent a risk for users, so noise information is not showed. Agents can not only adapt their UI (through changing states, since each state has a UI), but also cooperate with each other through joint intentions, dropping non-important or non-achievable goals (Section 4.5.1).

Nevertheless, comparing to the literature on SAW agents, autonomous SAW-driven interface agents are not trying to “understand” the situation, but assist users in understanding it. They use the human as a processor, supplying bits of information on all three levels of SAW to assist humans develop their own SAW. Humans are the final higher level “agent” solving the problem (of SAW acquisition and maintenance) by combining results from each agent. Even considering automation (System) awareness, their goal is not to provide improved intelligence for agents’ autonomic actions, as done in many projects [Mostafa13], but to provide information for the user to be aware of automation and to support human decision-making.

Actual examples of agents are given in Figure 4.12. Any agent belongs to/is a representative of an Awareness (or situation aspect, as defined in Section 4.2). For instance, an agent of Equipment Awareness is, by logic, an equipment. An agent from Procedure Awareness is a procedure, one from Environment Awareness is an environment, one from System Awareness is a system, one from Team Awareness is a team, and one from Enterprise Awareness is an enterprise. Although not explored in this thesis, a Personal Awareness agent represents a person.

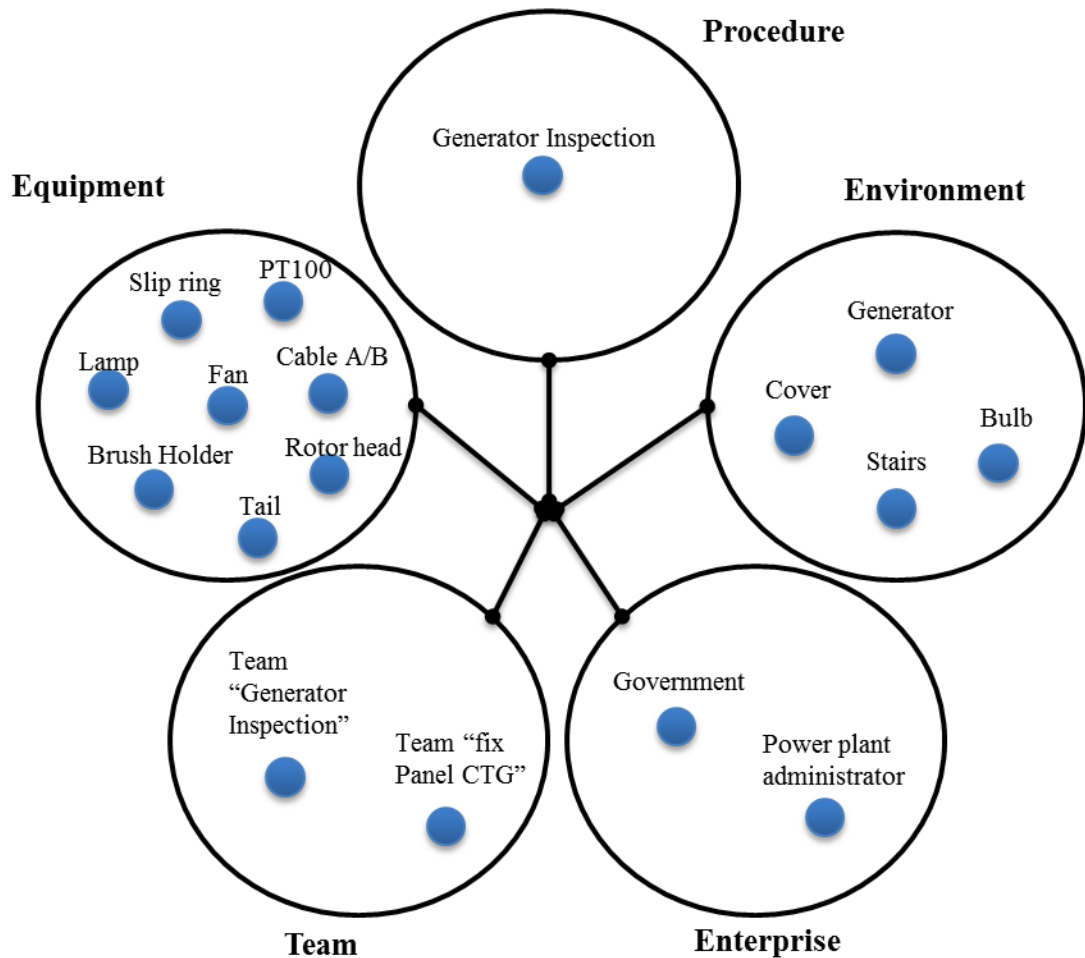


Figure 4.12: Example of agents and the network formed by agents and situation aspects (awarenesses).

Figure 4.12 also makes clear the distribution of awareness through the system, hence it is also compliant to the DSA theory. The network formed by all the agents is similar to a Distributed Situation Awareness graph (a propositional network, which is a knowledge network). DSA consists of an analysis of the whole system and in this case, system means the complete interrelation and sociotechnical interactions of users, artifacts, and environment, henceforth named as the “complete system”.

Nevertheless, autonomous SAW-driven interface agents are focused on one user, or **the** user (even if this user has a team). Instead of trying to distribute awareness among “agents” (human and artefacts), awareness is distributed among agents (all computational) to serve a user, since, as discussed, this thesis adopted a MAS architecture. Thus, although with a different focus from DSA, it replicates the theory (much like [Villaren12]), but with a focus on each user, constructing an awareness network for this user, instead of focusing on an awareness network for the “complete system”. This is supported by Stanton et al. affirmation that DSA can indeed be used in a single user system [Stanton06].

Following a principle of DSA, the focus is not on the nodes (the agents), but on the links (communication pathway). Links of the type agent->user are the UI generated by each agent. Links of the type agent->agent define the cooperation among agents. Knowledge (kept by agents) is not the central core, but the links are, because their activation (knowing which link to use, whether agent->user or agent->agent) is what determines the end quality of the UI (in this case, the user potential to acquire SAW).

This view of the combined theories of SAW (Endsley 1995 model of SAW and DSA) in the autonomous SAW-driven interface agents is presented in Figure 4.13 and Figure 4.14.

In Figure 4.13 a microview of the system side is presented, and four conclusions can be drawn from observing it:

1. Three autonomous SAW-driven interface agents are participating in this scenario. They could be any agent from Figure 4.12, but in this case they are abstracted as the greek letters γ (Gamma), δ (Delta) and ϵ (Epsilon). Each of these agents will support users by modeling decisions (in this case only one decision) by applying knowledge rules to raw data and transforming it into an output (the UI);
2. Inside each agent the three levels of SAW are modeled represented by three overlapping boxes. Level 1 information is presented by symbols (\otimes , \otimes , \otimes). Level 2 and 3 are represented as a capital letter (A1 and A2). For agent γ the level 1 are \otimes , \otimes , \otimes , level 2 is "A1" and level 3 is "A2". Thus the **current** situation of agent γ is "A1" and its **future** situation is "A2";
3. The level 1 data inside all agents are using the data structure from the FSA;
4. All agents combined are forming a propositional (knowledge) network, the approach established by DSA theory. This is possible because of the MAS architecture.

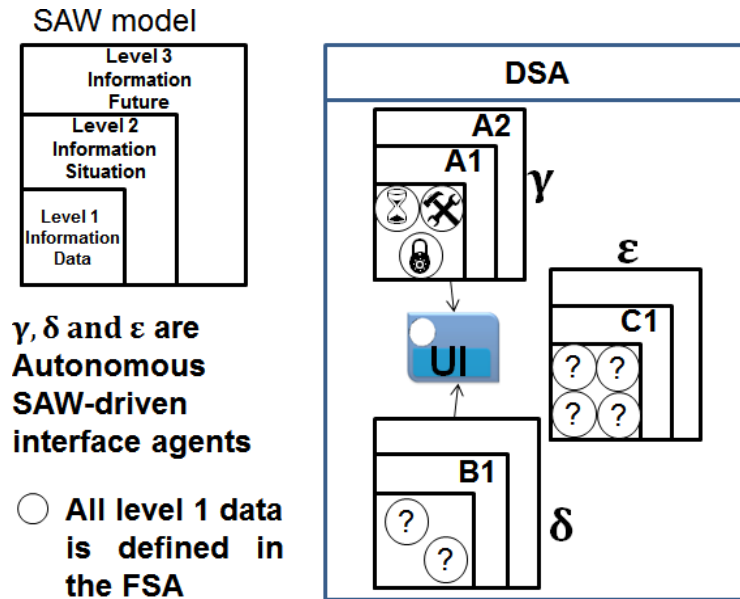


Figure 4.13: The view of the combined theories of SAW (Endsley 1995 model of SAW and DSA) in the autonomous SAW-driven interface agents.

In Figure 4.14 the evolution, in three periods of time (T1 to T3), of a situation is represented by the perspectives of the system and the user. The system side uses the concepts defined in Figure 4.13.

In the user side, an evolution of the user SAW (represented at each time frame as 3 overlapping boxes) is exposed. During each time frame user's SAW can be formed before, after or during the system updates. In many cases SAW is acquired by the user before the system shows any information, and the system acts as a confirmation. In other cases SAW is acquired because of the system.

Combining both sides, Figure 4.14 shows how the agents are feeding the user's SAW. At T1, agent γ displays 3 levels of information directly in the UI, and agent δ displays only level 2 information in the UI, so that the user knows every information from agent γ but only the situation (level 2) of agent δ . Both agents assist the user in forming an initial awareness of the situation (a "picture" of the situation). At T2, the level 3 information "A2" (future situation) became a level 2 information (present situation) and agent δ foresees the level 3 situation "B2". However, agent δ is **currently** not on display on the UI. Therefore in T2 the user still thinks the situation of agent δ is "B1", and at the same time starts to project (by him/herself) the future situation "A3", but still needs confirmation because information "?" is missing. This clearly demonstrates that users have their own SAW developing process and do not depend only on the UI to form an accurate SAW (after all, they are in the environment, living these situations). Finally in T3, agent γ projects a level 3 situation "A3" using δ and ϵ information (their level 2 "B2" and "C2"), causing it to also activate agent ϵ to be displayed in

the UI. In the final time frame for the user (T3), information “?” was revealed as “C2” and an accurate SAW was developed so far, for this scenario.

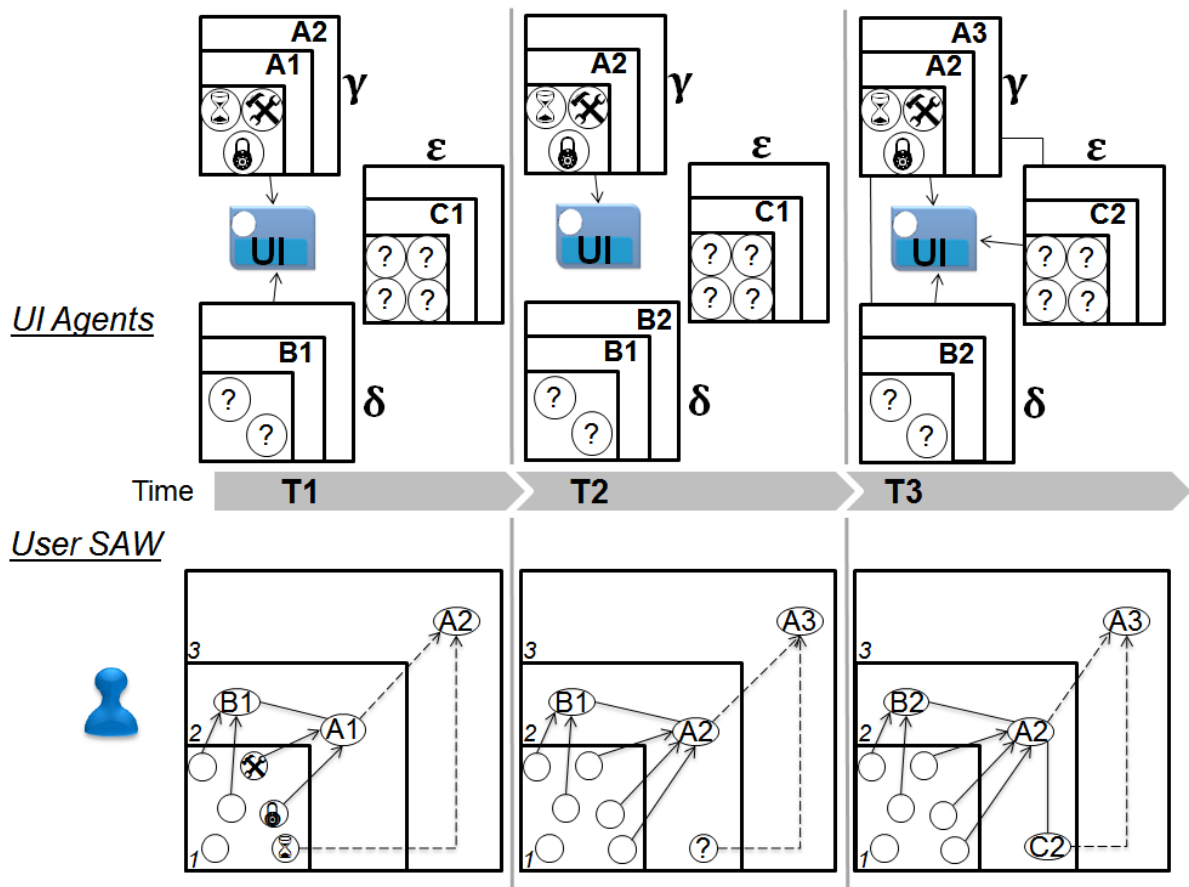


Figure 4.14: A UI assisting the user to develop his/her SAW during three time frames. In the UI part, every concept developed in this thesis is explored: Endsley 1995 model of SAW, DSA theory, FSA and autonomous SAW-driven interface agents using a MAS architecture.

The system architecture that allows the use of these agents, which partially combines Endsley theory of individual SAW with Stanton theory of DSA is explained in the next section [Endsley12, Stanton06].

4.5 Model-based run-time environment architecture for SASUI

The model of Situation Assessment defined for this thesis includes: the Saw-driven UI agents, with their abilities to express the situation and projection of an entity in the UI; the MAS architecture for distributing the process of Situation Assessment of the system, using the Saw-driven UI agents and their ability to communicate (for cooperation). This section describes this MAS architecture.

The multiagent SAW architecture proposed is a high-level architecture for a decision support visualization system, centered on assisting its user to acquire and maintain SAW with data provided through autonomous SAW-driven interface agents (modeled by domain experts) from support systems (i.e. E-Maintenance, Internet of Things, Wireless Sensor Networks). It is an architecture made to implement a SASUI in maintenance, which helps provide a uniform presentation for information in these UIs.

This agent-based architecture is based on the agents specified in Section 4.4. Agents are selected in run-time based on the user current goals, reinforcing the objective to create a SASUI. Their goal-driven behavior is created through triggers that are assigned to them by designers during their modeling phase (Section 5.5.3.2).

The architecture has the following features: agents acting to distribute cognition; agent reuse; a dual run-time environment separated as a virtual and local layer; agent independency and organizational model; partition dynamism and layers of information; agents easy access to each other and componentized data structure.

To explain every feature, a list of architectural views, inspired by the 4+1 model of Kruchten, will be offered [Kruchten95].

4.5.1 Agent acting to distribute cognition and for cooperative problem solving

This architecture distributes cognitions among agents, but from the point of view of individuals instead of systems. The cognitive demands for collaborative work (collaboration through teamwork and artefacts) in procedural tasks is different from command and control, hence the difference in view and use of DSA.

As stated by Stanton et al. “the ‘ownership’ of this knowledge is initially at the system”, and it remains for this thesis, as agents are the system [Stanton06]. Nevertheless, contrary to DSA, the focus is on the individual, because the focus is on UIs for each individual and not on the system itself.

In other words, it is less about the “complete system” synchronization to solve the problem and more about agents (the system) synchronization to empower the human with SAW so that he/she solves the problem.

This architecture can also be used to assist providing a synchronized “complete system”, if every single agent related to the system is modeled (every user, team, procedure,

equipment, environment, enterprise and system). Thus, the architecture can implement Endsley SAW model and DSA model.

To promote an effective distributed cognition, agents use a Cooperative Distributed Problem Solving (CDPS) strategy. The problem to be solved is the user acquisition and maintenance of SAW. CPDS recommends the use of three stages, which are implemented in this architecture using the following strategies:

- Problem decomposition and allocation: this stage is responsible for decomposing the problem into smaller and solvable subproblems. For this thesis, the problem was decomposed as the subproblem of having comprehension of each entity (agent) current and future state, considering that agents should use not only internal but also external data (from other agents) to solve this subproblem;
- Subproblem solution: this stage is when subproblems are solved locally, but cooperation among agents is fomented. For this thesis, each agent uses their Decision Component to define their states (which may use internal and external data), and they may share this info with others agents;
- Solution synthesis: this stage is responsible for assembling subproblems solutions to provide a complete solution. For this thesis, each subproblem solution is displayed in a UI, so that a complete solution can be devised by the user, who is the one responsible for taking decisions and acting in the system.

To better coordinate agents for the CPDS, a Joint Intention approach is implemented. Agents have a Joint Intention when they have a common goal and cooperate towards it.

All agents commit to all goals specified by the SME when designing them. They have a joint commitment to fulfil these goals. However, whenever an agent discover that a goal is redundant, because it is either already achieved, or irrelevant (there is no longer motivation to achieve it) or is unachievable, a convention forces the agent to drop the commitment and also tell the others.

Examples of redundant goals are: a Procedure was executed, thus the goal of accomplishing this task was already achieved; the user has left a dangerous environment, thus the goal of being safe in this environment is now irrelevant; there is not enough pieces to repair an equipment, thus the goal of repairing it is currently unachievable.

These commitments and conventions are encoded through a rule-based system. More specifically, conventions are implemented through the Messaging mechanisms of the architecture.

Agents have a DFA to define their level 2 (current) and 3 (future) situations (states). Whenever a transition occurs (in either level), a Message schematic can be defined for this transition. These messages currently (the set of messages can be expanded) permit passing information to agents, hiding of agents, switching the current procedure, and others. Using these messages, it is possible to promote coordination among agents.

Figure 4.15 shows an example of state transition and message. In this case, the current state (level 2) of an equipment was modeled. If the equipment is defined as ‘Normal’, then for transition **p1** a Message schematic was defined communicating to the agent responsible for the procedure of repairing the equipment, to warn this agent that the equipment is ok and does not need repairing.

To clarify, a Message schematic is the configuration of a Message mechanism to a specific case. Message mechanisms are the types of messaging possible (the complete list is given in Section 5.3.2). A Message schematic usually involves a mechanism, target, and other configuration parameters.

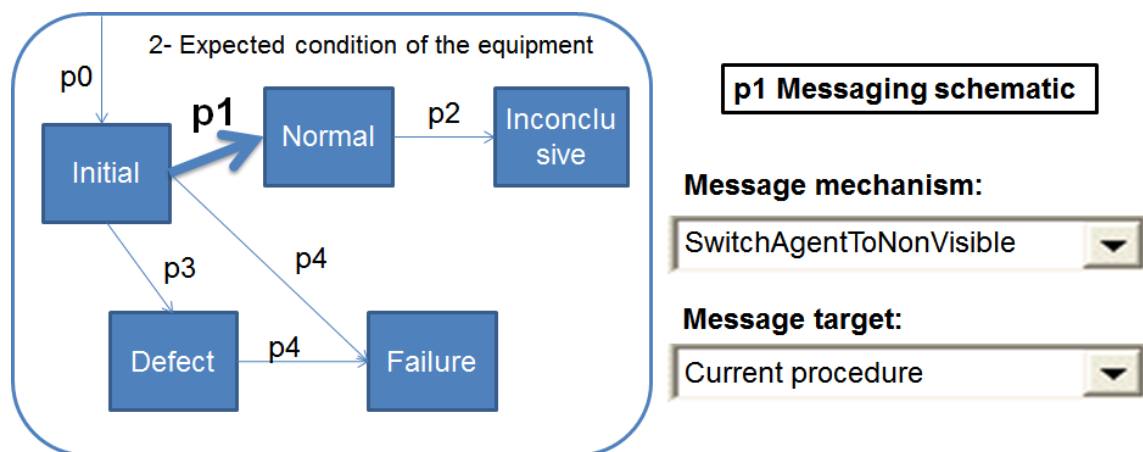


Figure 4.15 : an agent DFA with Transitions and using Messaging in an agent.

4.5.2 Agent reuse

One of the main goals of this agent-based architecture was facilitating reuse of UIs created per task. In Figure 4.16 this reuse is explored.

Each task has a set of decisions. These decisions are modeled inside agents, in their Decision Components. Decisions can be modeled by 0 to n Decisions Components of an agent, and a Decision Component can model at least one or even more decisions together. As decisions are being pulled out of the decision pool and modeled by agents, the agent pool (the set of all agents created and modeled) starts to grow. When all decisions are finished being modeled, the next task is modeled. This process is repeated until every task is modeled.

At any time, a previously modeled agent can be “reused” by being appointed as responsible for a decision. This happens because many decisions are the same among different tasks.

Even more, many agents are reused among different tasks. An equipment may be a part of several tasks, thus the representing equipment agent would be present in these tasks. Environments are basically the same (once they are all modeled) among all tasks, as are enterprises, systems, and sometimes even teams (although teams are highly dependent on the current task).

Therefore, as decisions are being modeled inside agents, the designer (the domain expert) job slowly goes from modeling new decisions, to choosing an agent that already has the decision modeled in its Decision Component.

This is a really important feature because enterprises may have hundreds of tasks to be modeled, and remodeling every exact same decision in the exact same agent would be time consuming and cost inefficient.

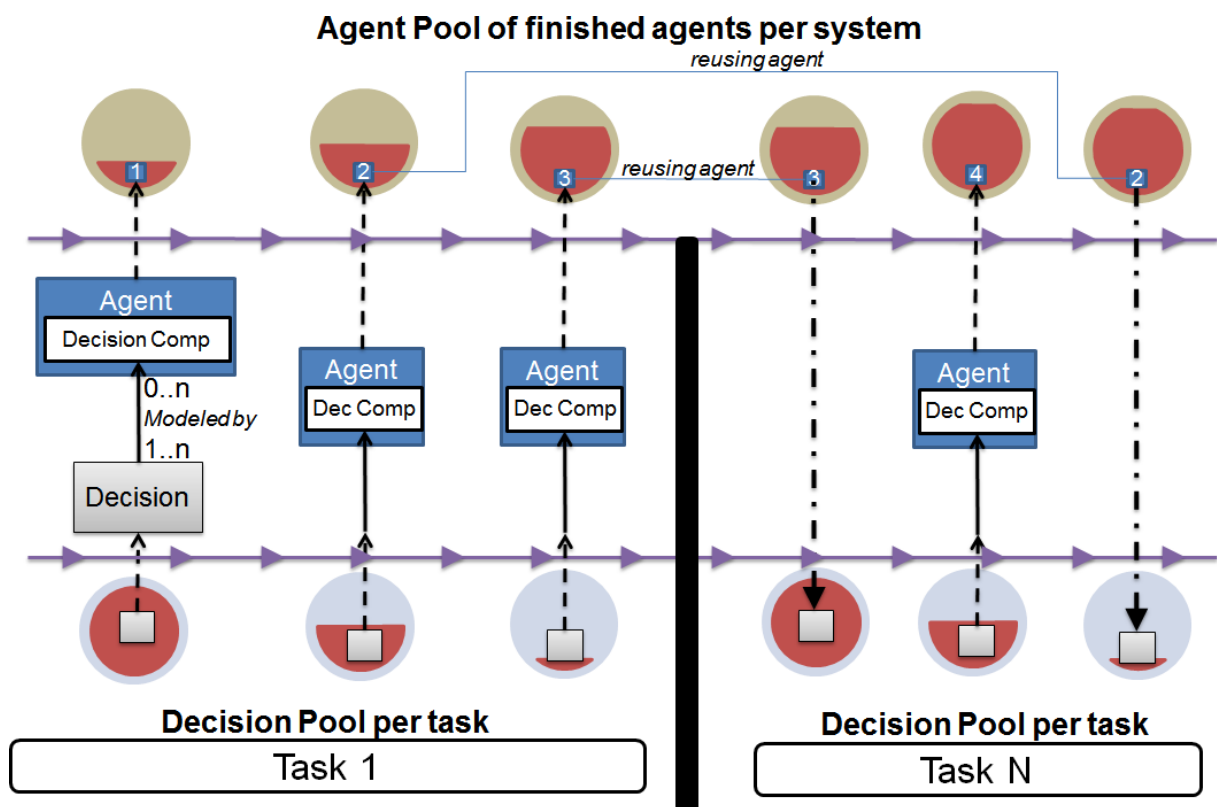


Figure 4.16: Architectural view of the reuse of agents in the system.

The term “Agent Pool of finished agents per system” is used in Figure 4.16 to denote that, while most agents are pre-implemented (the already discussed Existing Agents) for their necessity of unique interfaces with the E-Maintenance system, they initially do not have

decisions. As tasks are modeled, decisions are inserted in each agent and the amount of “finished agents”, ready to be reused, grows.

In future iterations of this project, if E-Maintenance (or other System of System) have their interfaces and APIs clearly standardized, automatic agent creation (and their Data Component) will be possible, requiring much less work and having more flexibility.

4.5.3 Dual run-time environment: a virtual and local layer

To facilitate the use of agents during run-time, they are separated in two layers: virtual and local.

Every single agent modeled is present in the Virtual Agents Layer. This layer can be seen not only as a container for agents, but also as a middleware between agents and the E-Maintenance system. This layer acts as an authoritative server [Bernier01].

Whenever a user starts an application and agents are requested, they are extracted from the virtual layer, and a local copy of the agent (a local instantiation) is made in the application.

Figure 4.17 shows this process, demonstrating agents being cloned into several different applications that are recognized as different tasks performed by different users.

Another view of this process is provided in Figure 4.18. Agents from the virtual layer are reused among many applications (local layer). Each agent can be reused by as many applications as desired and many indeed are.

For instance, consider an environment agent. If ten users enter this environment, the same environment agent will be used by these ten different users in ten different applications. Whenever an alteration in the environment happens, the environment agent communicates with each of the users that have a local copy of this agent (authoritative server).

In Figure 4.18, light colored agents (in virtual and local layer) are the same and connected by a client/server architectural pattern. The other agents in the local layer are also connected to a counterpart in the virtual layer, but the figure does not show this connection for the sake of simplicity and cleanliness.

The standard used to implement this feature is OMA Games Services Client Server Interface, from the Open Mobile Alliance (OMA) [OMAGame Services Architecture 11].

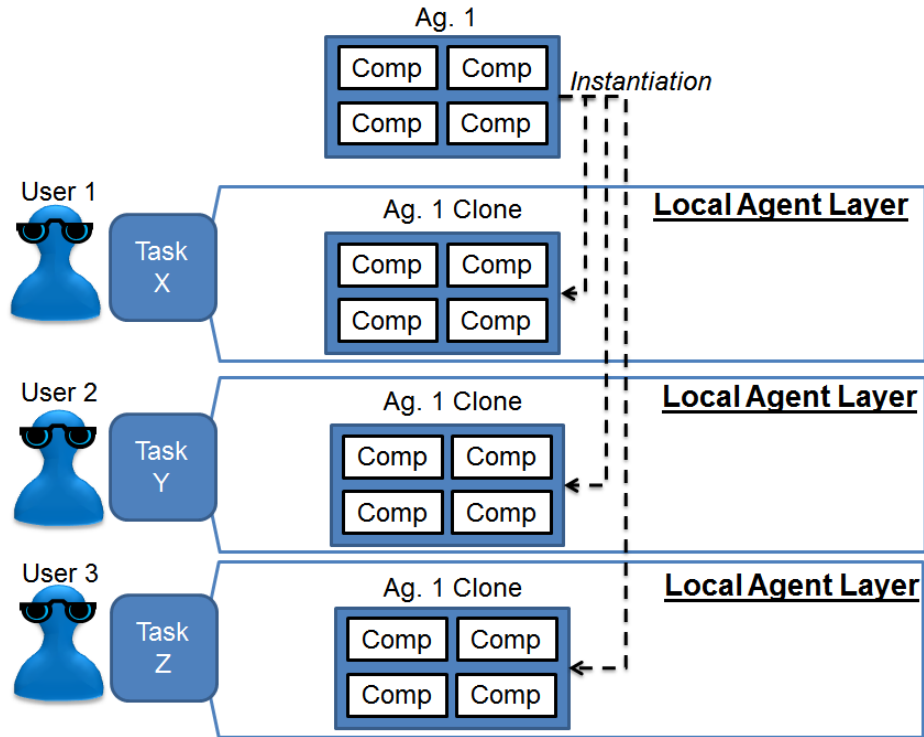


Figure 4.17: Logical architecture: scenario of agent instantiation.

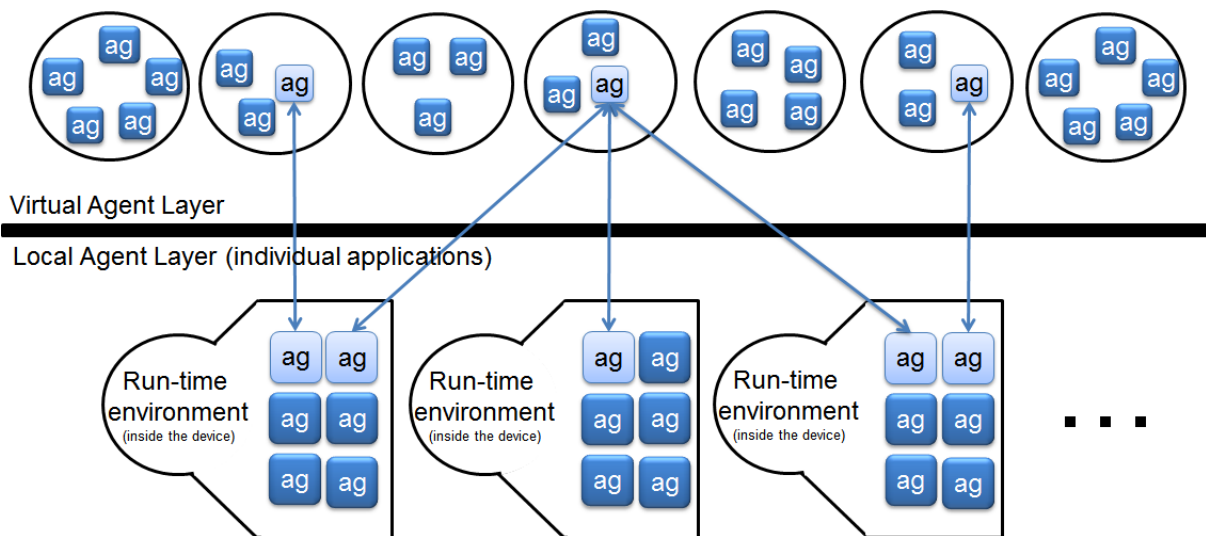


Figure 4.18: Process architecture: scenario of agent communication between the local and virtual layers.

Figure 4.19 reinforces this view, by showing the development view of the architecture. This architecture is based on 4 layers: system, virtual agents, local agents and UI.

Agents run on a centralized virtualization of the physical environment. This is because there is no need to actually instantiate an equipment agent inside an equipment, it is easier and more reliable to have all agents in a virtual layer to avoid sensor/IoT limitations (processing power, loss of connectivity, battery challenges, etc). In this virtual layer, agents from any type

(from any situation aspect) can exist and maintain constant communication (mostly request of information) with the E-Maintenance or IoT system.

Each instance of an agent for a user (maintenance technician) is a copy of an agent in the virtual layer (e.g. cloud). Therefore, the application has a Local Agents Layer with duplicates of agents running in the Virtual Agents Layer. This is because these local agents are specific for their user (owner), thus only being able to change their UI.

Finally, the fourth layer is the UI, which receives output from agents divided in Global and Local SAW, and renders a UI element with the modeled information in the screen.

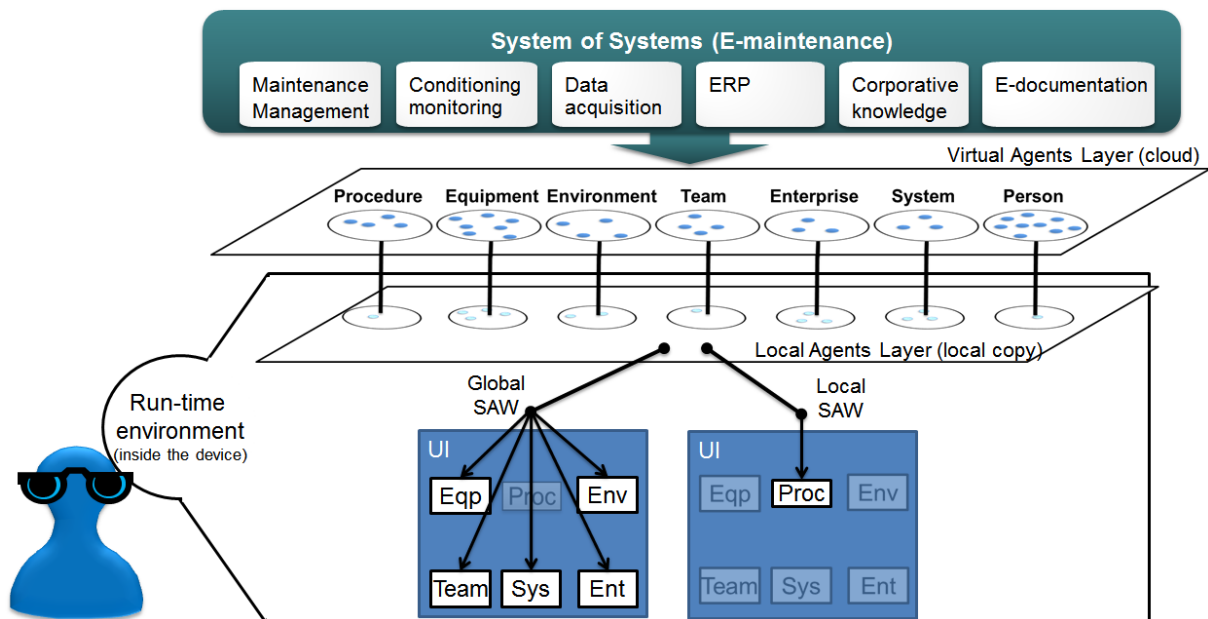


Figure 4.19: Development architecture based on 4 layers: system of system, virtual agents, local agents, UI.

4.5.4 Agent independency and organizational model

Every agent is independent (has to be fully functional by itself), but is connected to a parent Awareness, which is an aspect from FSA. This independence allows for an agent modeled for a task to be reused in other tasks.

The agents' hierarchical structure means that parent Awareness can be used to find information about every agent they coordinate. For instance, if a Procedure agent needs to find information about the current Environment agent, then it just needs to ask the Environment Awareness agent for the agent representative of the current environment. Using this method, any agent can find information in real time about the current procedure, environment, team, etc.

In a more formal definition, the organization model of the MAS architecture is of the type modified hierarchical, demonstrated in Figure 4.20.

Agents can communicate with one another freely to request information or abandon a joint commitment (drop a goal or a specific decision).

Considering most agents will need information from peer agents but they do not know from whom to ask, agents also communicate constantly with Awareness agents to request the agent that holds the information they are looking for.

In the modified hierarchical model, agents are also in communication with the master agent, which in this case is a layout manager. This master agent has the global view of the UI and is responsible to provide global optimization. In this case, this means optimizing the generated UI for the user (since this is the product from the agents).

Finally, another master agent, in this case responsible for global decisions, is the user (maintenance technician). They are the ones that will aggregate results from every agent and take the final decision to interact with the “complete system” to execute their job. Users can also promote/demote agents, acting as their own layout manager, choosing what information they are or aren’t interested at the time. This feature is further explained in Section 4.5.5.

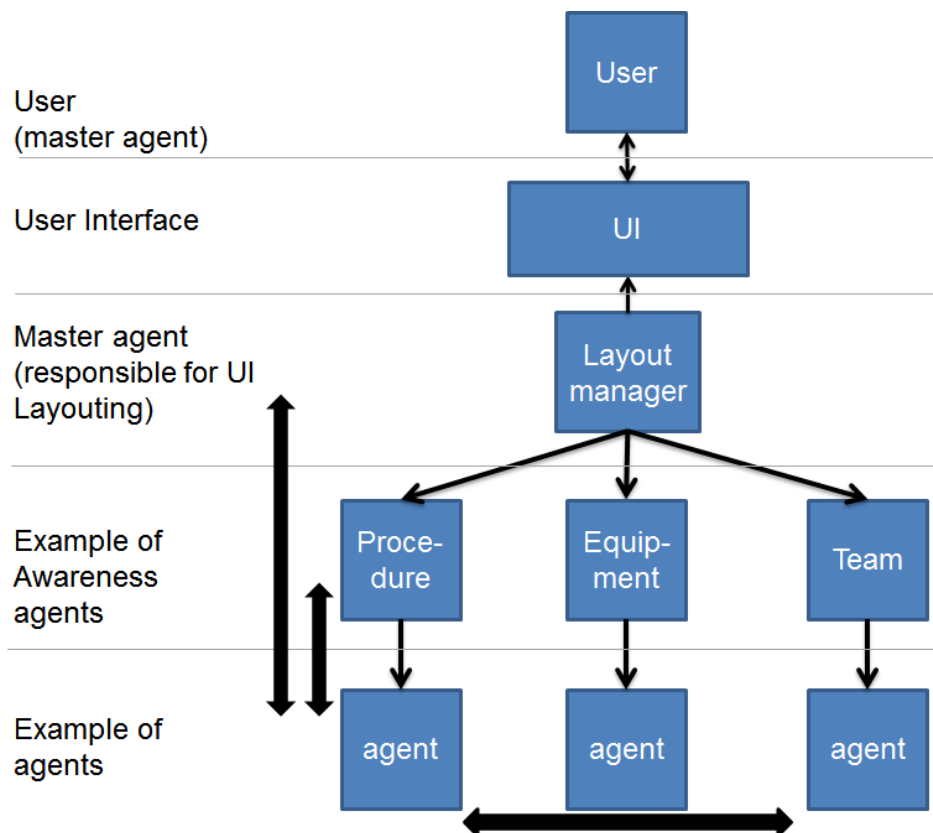


Figure 4.20: The organization model of our MAS architecture, a modified hierarchical model, allows for: direct agent communication; agent to awareness communication; both master agent to optimize (layout manager) and decide (user) based on a global view resultant of the observation of all the agents.

Besides finding and referring the correct agent, Awareness agents are also responsible for providing high-level basic functions to be used by their descendant agents to optimize calculations of certain advanced problems (specially to calculate levels 2 and 3 of SAW). Other projects also resort to this resource [Feng09], but our differential is that we are not limited by it (SMEs can define their own Situation Assessment rules).

Some of the high-level functions to be used by the agents are:

- Date: calculate date and time related operations;
- Spatial operations: calculate space-time based operations, like how much time is the user inside an environment, how much time does it take to move from point A to B, is the user moving, where is he/she moving to and how long will it take;
- Equipment reliability: calculate equipment reliability based on mathematical models;
- Procedure estimated time: estimate the time it takes to complete the procedure;

Finally, in some cases, agents may need information from other agents that will only be calculated in real time. In these cases, the requesting agent will have specified that this information will come in real time and will wait for the information. The supplier agent will process this information when possible, and message it to the requesting agent.

4.5.5 Partitioned Dynamicity and Layers of information

Another consequence of the awareness division is the partition dynamism [Weld03] in the UI, as shown in Figure 4.21. Each awareness has its position of origin in the UI, and inside it, information will constantly change. New information can be added or removed from an awareness, and all others can resize to adjust in the screen and not use all the screen size.

Structuring information using agents, and having Local and Global SAW specified, also allowed for a feature called Layers of Information, in which there are two modes: Visible and Nonvisible mode. This feature created two layers of information, one where information was only to guarantee Local SAW, and the other to reveal all available information for users about their current situation (Global SAW).

Tasks are started in Visible mode, where most information is hidden (only procedure is shown). By pressing an input button, the mode changes to Nonvisible, where users can see all information available for their current situation. Figure 4.22 shows an example of Visible (A) and Nonvisible mode (B) from the study case defined in Chapter 6.

The creation of these two layers of information also serves another purpose. In DSA, there is an assumption that if an agent holds the information then the user does not need to

know it until the right moment (not necessarily always, but is an important presumption from DSA theory). While this may or may not be truth (argument for both cases in [Endsley15, Stanton15]), for this thesis it is assumed as a truth for the type of work targeted (maintenance work). As does Endsley theory stating that SAW is not on the working memory [Endsley15], thus, after the user obtains SAW, it will not “fade” or be limited by working memory constraints. These two layers represent both concepts, because information considered important is included in the Visible mode, and all other information remain on Nonvisible.

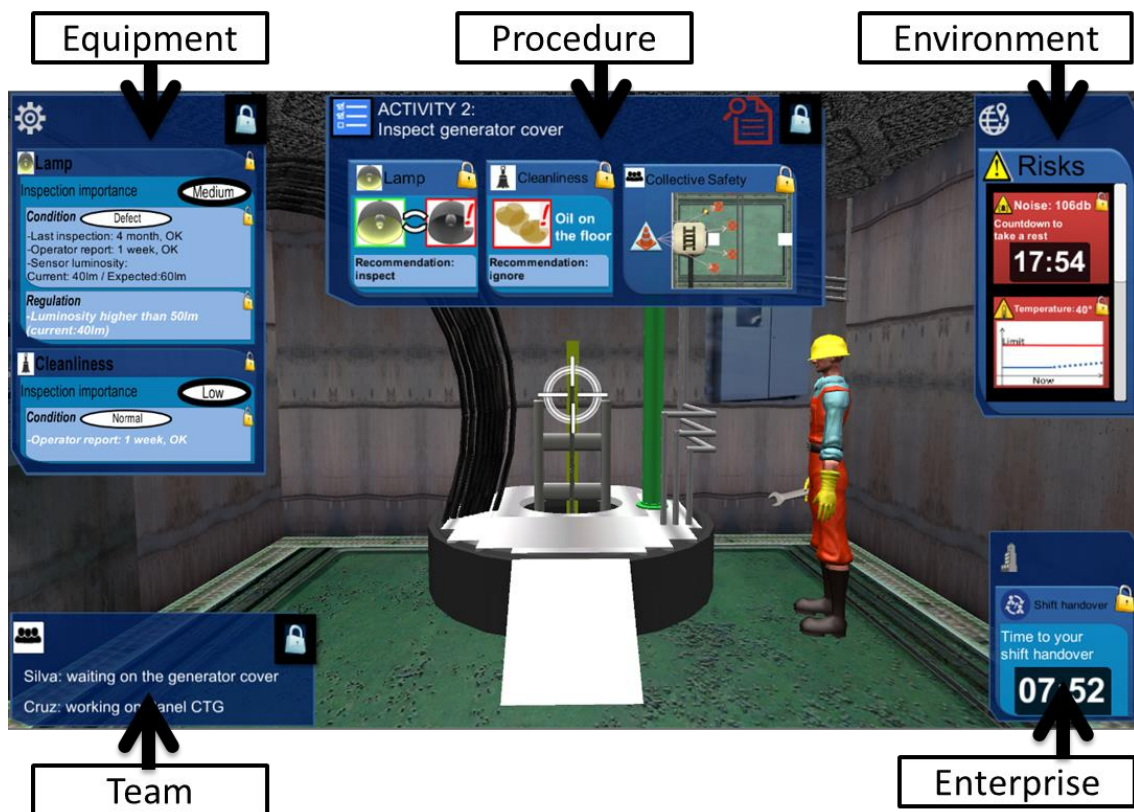


Figure 4.21: Partition dynamism, due to the awareness division, creates a clear division in the layout to characterize types of data in fixed positions for users.



Figure 4.22: The two visualization modes of the UI. In (A) the Visible mode show important information, usually Local SAW (procedure) and new information and events (plus information personalized by the user). In (B) the Nonvisible mode shows all information available and relevant for the current situation.

These concepts about information held in agents and SAW and working memory were also combined with Patrick and Morgan theory of cost of access [Patrick10]. Any information, at their first appearance, are included in the Visible Mode (and displayed to the user), but if the information was classified as not vital (non-important), after a short period of time it is relocated to the Nonvisible mode (and is hidden from the user, although it can be brought back at any time by the user by a few steps, which leads to the cost of access).

The natural question, in this case, of “What is an important information?”, is the exact purpose of why decisions are modeled in agents. Agents have a representation of the user SAW (Decision Component) in them, and they calculate the comprehension and projection (levels 2 and 3 of SAW) of their own situation (based on the modeled Decision Component). Each agent model is responsible for including agents in the Visible Mode whenever a projection of a situation becomes important for the user. Thus, in terms of DSA, this knowledge model provides activations of the propositional network link.

These layers also allow user personalization and novice/expert differencing (customization) without loss of information. Users (maintenance technicians) can choose, at any time, to pass an agent from one mode to another (thus choosing to hide or reveal it), personalizing what information they want to follow in real time (adaptive UI).

Regarding adaptation for novices’ and experts’ users (experienced in the domain and the system), novices could work with the reduced information density of the Visible Mode, while experts could pass more agents to the Visible Mode, increasing information density (and theoretically increasing their SAW, both Global and Local, generating increased decision-making accuracy). While this is currently user controlled, in future iterations this feature could create adaptable UIs (that customize themselves).

4.5.6 Agent easy access to each other and componentized data structure

Another feature is the agent’s internal structure and messaging structure. Figure 4.23 shows a logical and process scenario of the architecture.

Figure 4.23 (B) shows that agents are made of components, and that an agent requires at least one component to exist. Figure 4.23 (A) shows that there are two types of components, data and decision.

Data components, as explained, provide interfaces to access the data from the System of Systems (E-Maintenance) and also have static data supplied by the SME during design.

Decision Components access the agent data through their Data Components. A Specialized Decision Component is a specialization of a Decision Component to solve a specific problem.

From Figure 4.23 (B) to (C) there is an example of agent communication. Agents can request data from each other, can send data to other agents and send messages for cooperation.

Figure 4.23 (C) details a Decision Component. These components are constituted of a decision (from the GDTA) modeled through DFAs for the levels of SAW requirements (2 and 3). Besides these DFAs, the Decision Component also has a representative UI. That is why for every Decision Component there is an arrow leading to the layout manager, because this is the Decision Component output (the UI).

Some Decision Components can also send modification messages to other agents. In Figure 4.23 (D), agents are receiving messages from another agent (a cooperation strategy, defined in Section 4.5.1).

An in depth view of Decision Components structure is done in Section 5.5.3.

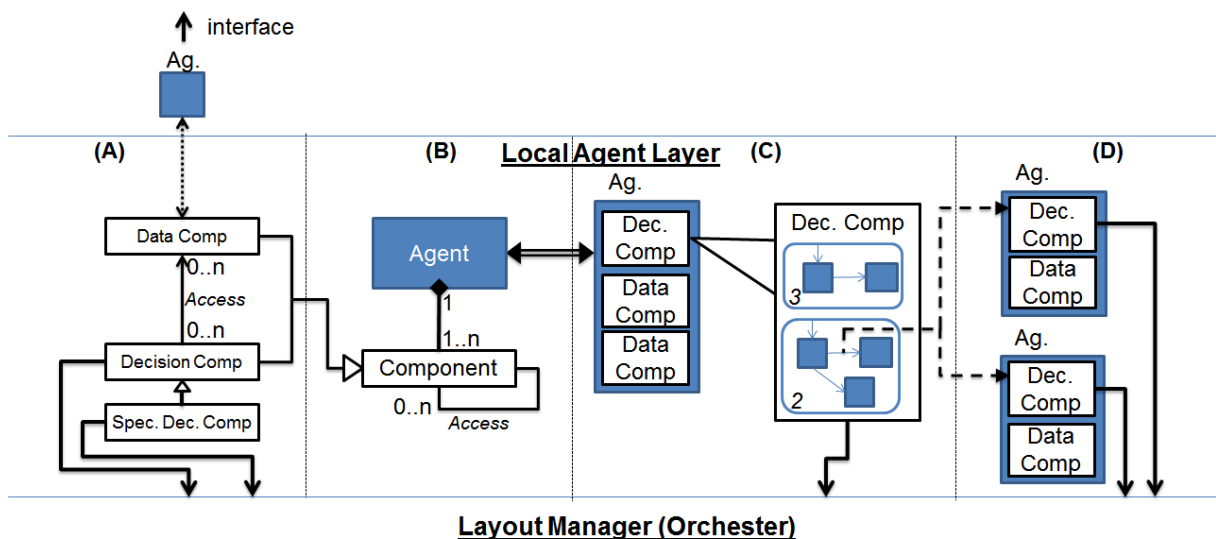


Figure 4.23: Logical and process architecture: scenario of agent execution.

4.6 Final Considerations

This chapter presented initial results obtained from this thesis. First the conceptual Framework of Situation Awareness Aspects (FSA) is proposed, as a result of the desire to structure SAW. This framework, although generic, was used to model a Situation Model for

the study case (maintenance), and from this results, guidelines and observations for UI design in maintenance were devised [Oliveira14, Oliveira15b].

Following, Local and Global SAW in maintenance work are discussed, establishing what they are and why they should be separated. This discussion will lead to an evaluation (Chapter 6) that analyzes the benefits of a UI that considers Global SAW.

Finally, the multiagent architecture of this thesis is presented. Initially the agents (autonomous SAW-driven interface agents) that compose this architecture are discussed in depth, with not only a characterization of these agents, but also their model and adherence to the fundamentals of SAW. Then the architecture is defined, with many views to explain its execution and how it allows a user (maintenance technician) to develop SAW by the assistance of the agents.

However, this architecture does not function if the agents are not previously defined by experts in the domain and carefully designed with the assistance from UI designers.

Therefore, the next chapter presents a methodology to specify the agents, consequently designing and developing the SASUI.

Chapter 5

MBSAW-UI: A MODEL-BASED DESIGN AND DEVELOPMENT METHODOLOGY FOR USER INTERFACES THAT SUPPORTS SAW

5.1 Initial considerations

In this chapter the main contributions of this thesis will be presented, a methodology called Model-based SAW User Interfaces (MBSAW-UI) used to design Situation Awareness Support UIs (SASUI) for maintenance work, which are instantiated through the architecture presented in the last chapter.

The current consistently used process to design SASUI is a high-level methodology composed from generic guidelines, called Situation Awareness Oriented Design (SAOD), presented in Section 2.3.2. There is a clear necessity to systematize SAOD, so that it allows a consistent creation of UIs that conform to the SAW demands of its users, and MBD is the technique fit for that, as already explained, because it can make the connection between high-level design processes and implementation frameworks.

Therefore, this chapter presents MBSAW-UI, a model-based design process that streamlined and systematized SAOD, allowing for consistent design of UIs that support SAW of their users.

First an overview of MBSAW-UI and all the stakeholders participating in the system development are presented. Then, MBSAW-UI is detailed, with a few examples.

5.2 Overview of the Model-based SAW User Interfaces methodology

The general goal of MBSAW-UI is to facilitate the use of the Situation-Awareness Oriented Design (SAOD), combined with others SAW-driven design techniques, by transforming them in a model-based design process and also adapting them for the domain of maintenance and industrial work.

5.2.1 Involved stakeholders

To overview MBSAW-UI, first the stakeholders should be discussed. Stakeholders are parties (groups) interested and partakers in the development and use of a system. For the type of system this thesis is addressed, these are the main stakeholders identified, with the type of target categorized according to the project scope [Gram97]:

- Programmer (implementer and system designer (interactive system designer, functional core designer); validator (software)): these are the personnel responsible for implementing the run-time architecture (Chapter 4). To be clear, the “programmer” category is a simplification of the many people responsible for software development (testers, QA, architects, etc);
- UI Expert (implementer (UI); validator (usability); system designer (UI); requirement specialist): these are responsible for UI and UX (user experience) design of the software;
- Industrial Engineer (client; user representative; end-user; SME): industrial engineers are experts on all parts of the industry. They understand maintenance, ergonomics (and safety), production, logistics, and others aspects of the industry;
- Industry Expert (user representative; end-user; SME): these are the major SMEs of the domain. They are coordinators, supervisors and managers that plan and control (and sometimes even execute) processes, like maintenance, of a plant;
- Technician (user representative; end-user; technical worker): these are the users of the application generated through the MBD process. The nature of their work revolves around manual labor, physical effort and immediate risks. The decision problem they usually face is classified as structured and operational [Vercellis09].

5.2.2 Model-based Situation Awareness User Interface

MBSAW-UI starts by developers doing a Cognitive Task Analysis, which are the GDTA and the cognitive map FSA (Chapter 4). From these analyses, agents and a SA model are defined for the system.

Then, the run-time architecture is implemented (followed the proposed in Chapter 4). To speed up implementation, developers of the architecture may use the FSA as a base (diagram class) to create Data Components and interfaces for Existing Agents³.

After the architecture is implemented, the UI of each task is modeled. For each task, participant (relevant) agents are defined, and decisions from the GDTA are associated with each agent. Each decision is then modeled (inside an agent) to consider, regarding the agent, several situations (level 2 SAW) and projections of situations (level 3 SAW). UIs are defined according to these situations and their projections, to assist users in acquiring and maintaining SAW.

All this process is divided in three stages: 1) Requirements Engineering; 2) implementation; and 3) UI Design stage, each with its own phases, demonstrated in Figure 5.1.

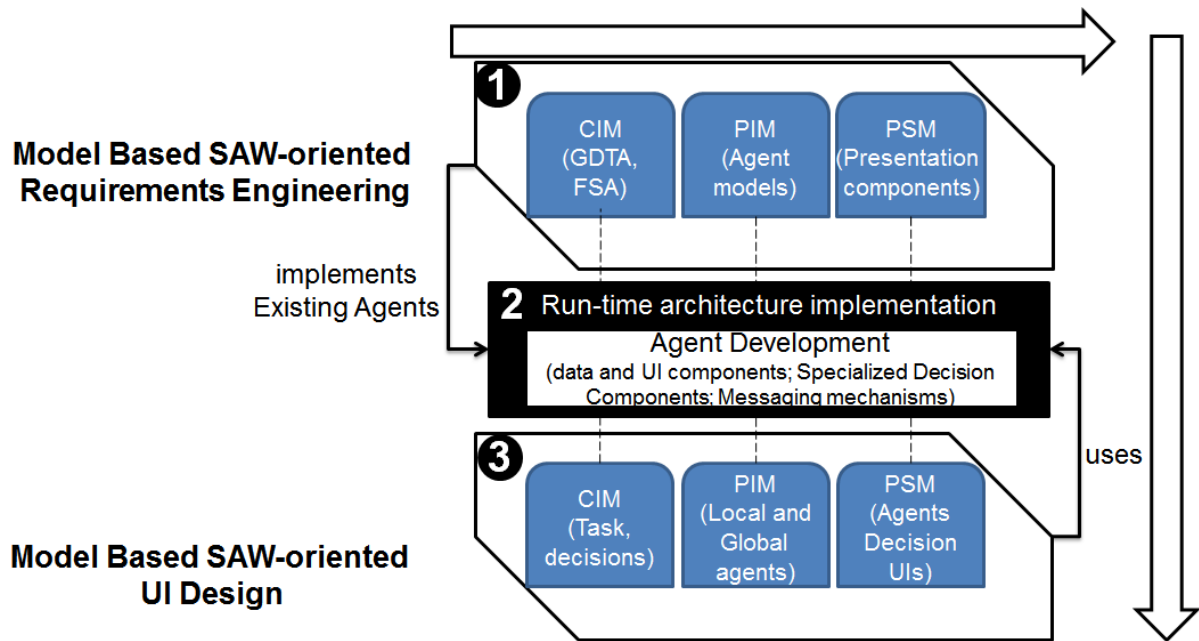


Figure 5.1: Overview of the methodology MBSAW-UI, for UI design and development to support SAW in maintenance work.

³ Existing Agents: agents that need to use information that is dynamic and thus should be connected to the E-Maintenance system.

The Requirements Engineering stage is subdivided in three other phases, which correspond to the MDA framework. They consist of: (CIM) elicitation, analysis and representation of requirements; (PIM) defining the SA model for the particular application (with the possibility to create new models for Decision Components); (PSM) designing presentation components for the UIs.

The first phase of the Requirements Engineering stage executes analysis to establish the cognitive requirements of the work. The techniques used in this step will both assist in collecting requirements and starting the design of the system. The results are a Cognitive Task Analysis (GDTA) and a cognitive map of the situations (FSA).

From these results, in the second phase, agents are identified and listed to be implemented as Existing Agents. Also in this step, the SA model can be incremented, with new models of Decision Components and Messaging mechanisms created. The Standard Decision Component (StanDC) is usually enough for most decisions, however Specialized Decision Components (SpecDC) can greatly improve development time for the UI Design stage. The results from this step are definitions of new SpecDCs and agent Messaging mechanisms, and a list of identified agents.

In the third phase, UI components for the system are defined. Most basic UI components are already included, but some specific components can be defined and implemented. The result of the Requirements Engineering stage serves as input for the next two stages.

The second stage is the development of the architecture defined in Chapter 4. To implement the core of the architecture, a framework is available. Developers only have to extend it implementing Data Components of Existing Agents (and their interface to the E-Maintenance system), and possible new Specialized Decision Components, new agent Messaging mechanisms and new UI components.

In the UI Design stage, the system is already implemented, and it is now up to SMEs and UI designers to feed it. In this stage, tasks based on the Work Instructions of the enterprise are defined. For each task, agents are created/defined to model decisions, using the components established in the previous stage (Standard, Specialized or Prompt Decision Components). First Local SAW agents, and then Global SAW agents have their Decision Components modeled, with DFAs, representation and messaging schematics.

5.2.3 MBSAW-UI in the software development process

Compared to a software development process, the first stage of MBSAW-UI implies analyzes and design, the second stage implementation, and the third stage is an EUD stage, with users designing incrementally the UIs of the system.

After the second stage, the system is not yet functional. It has the Existing Agents, but only in the third stage supported tasks and their UIs are defined and inserted in the system.

This division of responsibilities was established due to the constraints of the industrial domain. As already discussed, not only several tasks (hundreds, in most industries) should be supported by the system, but also the unpredictability of this domain demands constant support from system developers.

In this methodology, end-users have control over the system supported tasks (and their UIs), and they can constantly update and create new UIs and tasks. Developers support is only necessary when new entities are inserted in the plant (new environment, new equipment and new system), creating a new Existing Agent to represent this entity.

The result of the third stage is a design model (or instantiated model) that will be loaded by the run-time architecture inside devices, which will then instantiate agents locally (as discussed in Section 4.5.3), begin communication with the E-Maintenance system and be ready to guide users (maintenance technicians) through the modeled tasks (Work Instructions). As displayed in Figure 5.1, the first stage defines, the second implements, and the third uses the system.

Finally, Figure 5.1 shows a clear connection from each phase of the first and third stages. This is because they are tightly coupled and decisions from the first stage directly affect and even dictate their counterpart of the third stage.

This connection is explained as:

- CIM: the CIM phase from the first stage is responsible for modeling decisions (GDTA), the situation structure (FSA) and defining the list of main activities for the domain (e.g. repair, inspection, etc). In the third stage, stakeholders classify tasks according to the list of main activities, and then proceed to establish the decisions of each task, according to the GDTA. Therefore, the third stage CIM is guided by definitions from the first stage.
- PIM: in the PIM from the first stage, agents are identified, and to increment the SA model, new specialized decisions and Messaging mechanisms are established. In the third stage PIM, a decision is attributed to an agent (Existing Agent or newly created one) and this decision is modeled according to either the Standard Decision Component or a specialized

one. This means that the identification of agents and the SA model defined in the first stage is important for their use in the third. SpecDCs and new Messaging mechanisms are established in the first stage to be used in the third, and they can respectively speed up design and give more flexibility to the system.

- PSM: in the first stage, the PSM phase defines UI components, which are then used by agents to display decision results in the third stage.

To extend this overview, in the next sections, all stages will be thoroughly explored with small examples of their functioning. Chapter 6 presents a complete study case of the use of the MBSAW-UI methodology.

5.3 Model-Based SAW-oriented Requirements Engineering

In this stage, requirement for the future system will be gathered and modeled to generate a system and prepare it for the EUD process. Figure 5.2 shows all the phases for this stage, which will be explained individually in the next sections.

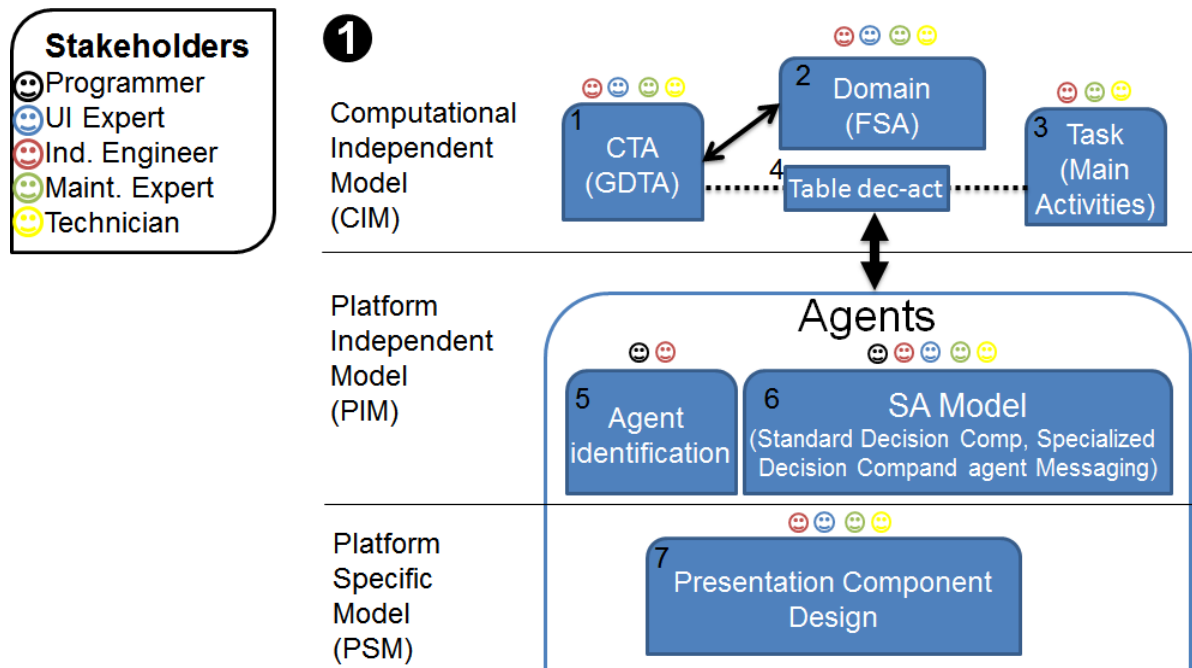


Figure 5.2: Model-Based SAW-oriented Requirements Engineering with stakeholders.

This methodology starts by identifying goals and decision-making information requirements, including how operators will integrate this information to form a higher level

understanding of their situation (levels 2 and 3 of SAW). Therefore the first step is a Cognitive Task Analysis (CTA).

5.3.1 GDTA and Framework of SAW Aspects (FSA)

Initially a Cognitive Task Analysis should be performed. This analysis is usually divided in three parts: elicitation, analysis and representation.

For SAOD the process called Goal-Directed Task Analysis (GDTA), which is mostly an elicitation technique, was deemed as sufficient. Complementary, a representation technique that can be used is Framework of SAW Aspects (FSA), which maps all the important information that constitutes the user's SAW in any domain.

GDTA is an analysis used to list goals, decisions and information requirement of users (Section 2.3.2.1). Using only a GDTA there is a vacuum on expressing, sometimes, what information is outside of main goals. This is because GDTAs can really focus on information users say they require to take decisions, but (especially for industrial work and maintenance domain) users forget they are constantly taking small decision outside the procedures.

Both should be created at the same time slot, so that they help each other advance. Sometimes information is first listed in the FSA and then a goal/decision in the GDTA is created using this information, and sometimes it is the opposite.

5.3.1.1 Creating a GDTA

To create a GDTA, first several interviews with SMEs have to be done. In this interview, questions should focus on goals and decisions the workers (future system users) make during work. Additional material, such as documentation (Work Instructions), can be used to improve the GDTA.

GDTAs focuses on dynamic information an operator need, to take decisions during task, instead of static knowledge of a system. For instance, a GDTA will not focus on eliciting from operators what is the exact routine of a procedure, but will focus on information required to execute this procedure and underlining decisions.

After the interviews, the first step is determining a goal structure. This structure is modeled as a hierarchy, with a major goal dividing into goals that divide into subgoals.

In sequence, decisions are listed for each goal. These are the actual decisions users will take during the task to fulfil a goal. For each decision, a list of information requirement is assembled. Information should be divided in levels of SAW (1, 2 or 3).

Creating a GDTA is a highly iterative process, meaning it will not be done in the first try. It takes time to perfect a GDTA, analyzing interviews to find new information and showing the GDTA to experts to request advice for improvements. In depth creation of a GDTA and tips for improvement can be found in [Endsley12].

Figure 5.3 shows an example of a GDTA considering a corrective maintenance (fix the equipment when it breaks). The GDTA was created for command and control operations, which has usually only a few activities that the user will perform. On the other hand, considering maintenance, there are hundreds of activities the technician can perform.

Therefore, to create a GDTA for such domains, it is recommended to create a generic GDTA that can be useful for most activities. This GDTA should have as most information as possible (they will be filtered during design). Appendix A shows a GDTA for the study case of maintenance, which covers corrective, predictive and preventive types of maintenance.

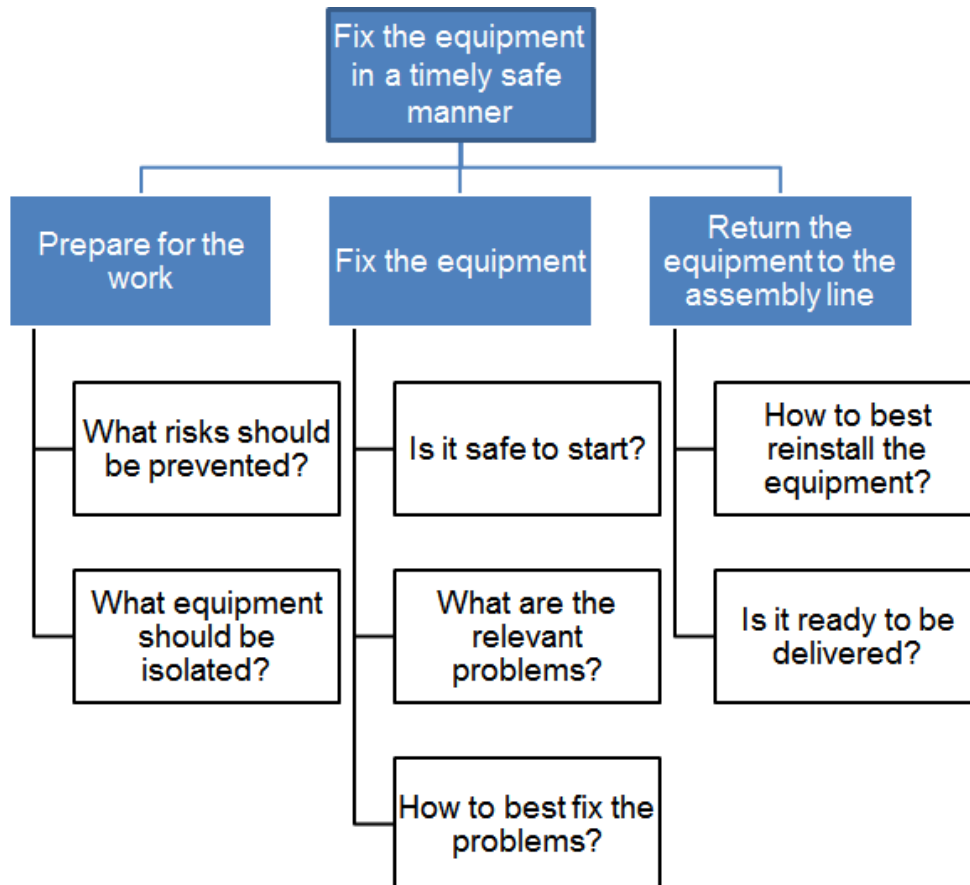


Figure 5.3: Example of GDTA for a corrective maintenance task.

Additionally, many times a decision (A) is linked to another decision (B), in a way that information requirement from the decision A also requires the same information requirement from decision B. In this case, instead of repeating this set of information, the keyword “callout” can be used to represent this repetition, like in the example below:

Decision: Which equipment and tool should be used to prevent personal and environmental risks?

Information Requirement:

Callout: “Decision: What are the risks I should prevent?”

2. Protective gear condition
 1. Location of protective gear
 1. Type of protective gear

In this example, having to decide of protective gear to equip involves also deciding what risks are important.

5.3.1.2 Creating a FSA

The FSA has three node types: Awareness, Data and Filters. Data entities are the information users require to execute their task. Filter entities are the information users do not need to know but that can be used to change the visualization of the Data entities. More in depth information can be found in [Oliveira14, Oliveira15b].

To create a FSA for a new domain, first the situation aspects (or awarenesses) have to be defined. The most important aspects of the awareness in the domain should come up from an investigation of the domain (interviews, documentation, white papers). It is probable that, for most industrial works (such as maintenance), the situation aspects will be the same as the ones listed in this thesis. Data from these aspects is what will change based on the domain.

Figure 5.4 shows an example of an FSA, showing the modeling of the Environment Awareness for maintenance.

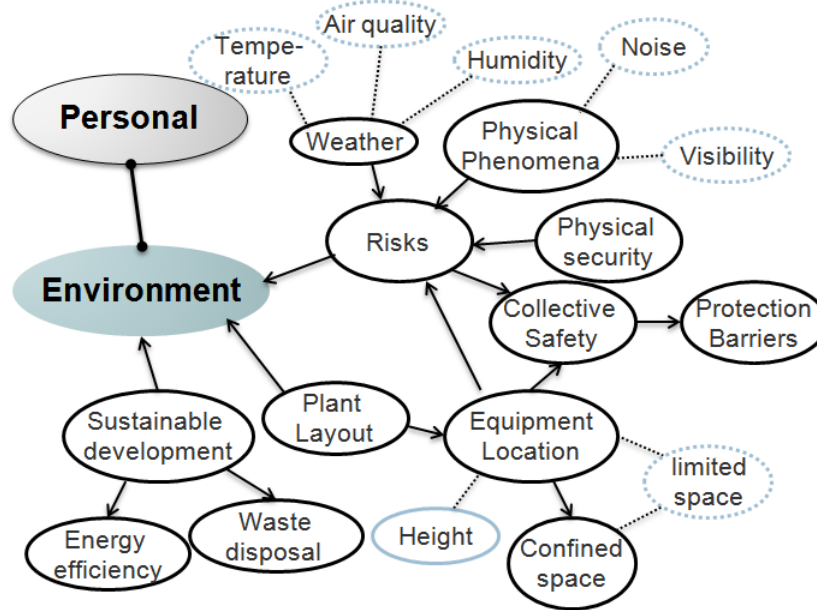


Figure 5.4: Environment Awareness in the FSA of the maintenance domain.

Next, Data and Filter nodes should be determined. Data nodes are basically data useful for the worker to perform the tasks and Filter nodes are data that may be used by the agents to improve their SA model and ultimately change the User Interface.

Data nodes should be composed through an iterative process, together with the GDTA. This exchange in focus, going from GDTA to FSA and reverse, will give experience to prepare a more complete Cognitive Task Analysis.

Data nodes are either a leaf node (no descendant), or an aggregation (i.e. A is composed of B, C and D), represented by a leaving arrow, or a specialization (i.e. B, C, D are special cases of A), represented by an arriving arrow.

Filter nodes are mostly information that will be used in states definition and agent messaging. Most of them are Human Factors data, like automaticity, user experience and stress. Even the ones from the environment will be used according to Human Factors studies, such as how sound (noise), temperature (cold, hot), air quality, vibration, and others, affect people during their work [Parsons03, Helander06].

5.3.1.3 Activities and the table of decision-activities

The third part of the process is listing and categorizing activities in the enterprise. With all activities listed, a table to correlate activities to decisions should be made. This table of decision-activity connections has three columns: 1) Activities, which discriminate all the activities defined, and also has a “common” section, for decisions that are present in all

activities; 2) Situation Aspects, to list if and which aspects of the situations is important to have awareness for the decision; 3) Decisions, listing each decision.

Table 5.1 is an example of a decision-activity table, considering **only** the GDTA built in Figure 5.3 and only **two** activities for maintenance: restoration and diagnostic.

Table 5.1: Example of a decision-activity table considering the activities of restoration and diagnostic and the decision from the GDTA of Figure 5.3.

Activities	Situation Aspects	Decisions
Common	Environment Equipment	What risks should be prevented?
	Equipment	Is it safe to start?
	Team	
Restoration	Equipment	What equipment should be isolated?
	Equipment	How to best fix the problems?
	Equipment	How to best reinstall the equipment?
	Equipment	Is it ready to be delivered?
Activities	Situation Aspects	Decisions
Diagnostic	Equipment	What are the relevant problems?

The GDTA, FSA, list of activities and decision-activity table are all artefacts that will be used on the third stage of this methodology (UI Design) to accelerate the design work.

After all these artefacts are produced, the next phase of the first stage commences, to identify agents and define the SA model.

5.3.2 Agent identification and Situation Assessment (SA) Model

The first step of this phase is agent identification, which is the same as entity identification. The recommended process to identifying agents is using the FSA as an igniter for the search of real world entities. FSA established that the aspects of a situation in maintenance (and most industry) are seven. Thus, agents are representative of these 7 aspects.

Thereby stakeholders should list procedures, equipment, environments, teams, enterprises, systems and persons. These will be the agents that will be implemented in the implementation stage (second) and modeled thoroughly in the UI Design stage (third).

The only imposition for this process is that any agents that use data from E-Maintenance system should be identified now or else they will not be implemented.

Table 5.2 shows examples of agents of all types:

Table 5.2: Examples of agents of all situation aspects (awarenesses).

Procedure	Equipment	Environment	System	Enterprise	Team	Person
Gather lubricating oil	Lantern	Bulb	Admin (SAP)	Company	Team eagle	John
Lubricate and clean PMGs	Slip ring	Cover	Maintenance (CMMS)	Govern-ment	Team alpha	Kurt
Maintenance and testing the speed regulator	Brush	Generator	SCADA	Regulatory Norm	Team beta	Luane

The second step in this phase is defining the Situation Assessment model. The SA model proposed in this thesis is defined by the combination of a MAS architecture with the autonomous SAW-driven interface agents, as described in Chapter 4. These characteristics are immutable.

However, the SA model can be improved by the stakeholders proposing two incremental features: new Specialized Decision Components and new Messaging mechanisms.

The default model of Decision Component, named as the Standard Decision Component (StanDC), is based on: a trigger to start the decision; state machines (DFAs) that represent knowledge of situations and projections of the agent (their entity); a UI; UI transformations; messages passed to other agents as a form of communication.

Therefore, whenever designers start to model a decision (in stage three), they will model DFAs, a set of presentations and Messaging schematics in a StanDC.

Initially, the architecture (Chapter 4) provided one type of Decision Component, the StanDC. If desired (or deemed necessary) by the stakeholders, new Specialized Decision Components can be created, increasing the set of models of Decision Components to StanDC plus all SpecDCs created.

These SpecDCs are useful to accelerate design by allowing to simply configure a set of parameters do define a decision (instead of having to define DFAs, presentation and

messaging schematics). Nevertheless, SpecDCs can only use the same resources to communicate outside themselves available to StanDCs, which is the Messaging mechanisms, and they also have to generate the same output, a UI.

Another type, PromDC, is basically a StanDC that is ready to be used. Since StanDCs are so flexible, each definition of them can generate widely different results. Therefore, whenever a StanDC is modeled to fit an agent decision, this modeled decision becomes a PromDC, which is a Decision Component ready to be reused by other agents.

Figure 5.5 gives an overview of this process, showing that SpecDCs are specialization of the StanDC, and PromDC are defined StandDCs.

In this current stage/phase, only SpecDCs are created. Two SpecDCs were created for this thesis as examples, to be used in the study case, and are presented in Appendix B.

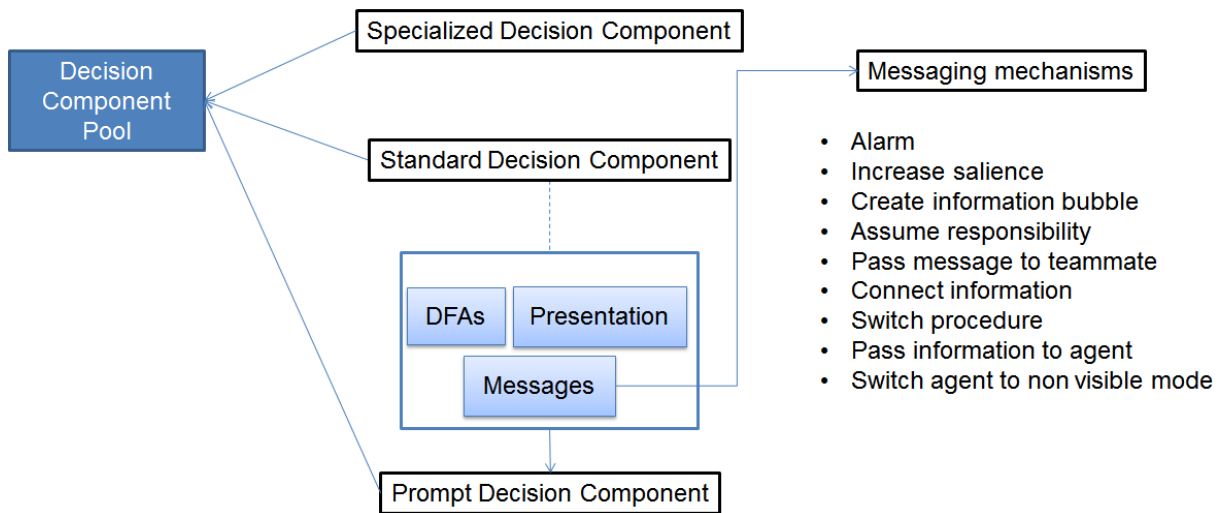


Figure 5.5: All the types of Decision Components, their structure and the possible Messaging Mechanisms.

Finally, in this phase the Messaging mechanism can also be incremented. As shown in Figure 5.5, currently these are the Messaging mechanisms implemented:

- Alarm: the mechanism of alarm is used to warn the user of a change in a situation, providing anticipatory feedback [Lieberman97]. The system response is sounding a sonorous alarm while flashing colors to increase the salience of the agent UI;
- Increase salience: similar to the alarm mechanism, however with more control from the designer, since this mechanism has a target, which needs to be another agent (and not the agent creating the message) and also has properties, which is the information field to be made more salient (it can be the whole agent UI or just one data field) and the duration of this action;

- Create information GUI: this mechanism creates a small text field GUI with information for the user. Its parameters are the position, duration and the text;
- Assume responsibility: this is used to inform the system the user is responsible for an equipment or a procedure;
- Pass message to teammate: this is used to communicate with teammates. It can be used to inform others that a task was completed, ask for assistance, to warn others of potential risks in their activity, to request solo use of an equipment and others;
- Connect information: this mechanism is important when a transition occurs that relates many different agents. Using this mechanism, the involved agents are visually connected, so that the user can better comprehend a link between them;
- Switch procedure: this is used when a more important procedure should be executed by the user, for instance when an emergency happens;
- Pass information to agent: this is used when an agent is responsible for informing another agent of a calculation result. For instance, when a system agent obtains information about the system and needs to inform the procedure agent;
- Switch agent to Nonvisible mode: this mechanism puts an agent in the Nonvisible mode (Section 4.5.5). It is used when the agent decision is out-of-context, due to either activity, time, risk, location or coworker's condition.

These Message mechanisms already cover a lot of possibilities. However, if required, stakeholders can create new mechanisms to give more flexibility to their systems.

Finally, in the last phase of the Requirements Engineering stage, there is the possibility of defining new UI components, explained in the next section.

5.3.3 Presentation components design

The same way that the previous phase allowed to extend the architecture, permitting to define new models of Decision Components (SpecDC) and Messaging mechanisms, in this phase, stakeholders are allowed to propose new UI components.

In many cases, these phases are not important and can be skipped, because the architecture is already flexible enough to build a solid system with a SASUI. However, having in the MBSAW-UI methodology opportunities to improve the architecture allows for unforeseen/unpredictable requirements to be included in future systems developed.

A list of UI components was already defined and implemented, but new components can be defined to increase the systems capabilities. These components can be easily created using the basic visual types, which are: image, string, number and line renderers.

UI components have three essential elements: a set of data input that lists all the data (and their types) that are present in the component; a processing script that defines how the input data will be visually transformed; a set of visual outputs using the basic visual types, that characterizes the visual format (layout) of the component.

Figure 5.6 displays the process that was used to define a UI component for a graphic of a variable evolution over time. In the left side, four input data are defined: the current value of the variable; the rate at which this variable is changing (trend); the unity of the variable (degrees, time, percentage, etc); the safety boundary of this variable.

Then, a script was implemented to specify how the three line renderers will work, one for showing users the past values, one for the trend and one for the boundary. These scripts are basically expressing how to update these line renderers (where to start and finish). More complex behaviors to create curves (quadratic Bezier curves) are currently not supported (they would have to be one of the basic visual types).

The final result of the output is then determined by using the variables defined in the data input, the new variables created in the processing script and the basic visual types. In the output, each UI component is defined with a type, a variable to determine if they are dynamic or static and the name of the variable to obtain the values from. An example, from Figure 5.6, is `LineRenderer:dynamic -> pastValues`, which means that a dynamic (loads data in run-time) line renderer will use the variable `PastValues` to render itself.

In the stage of UI Design (the last stage of the MBSAW-UI methodology), designers will use these UI components to define the graphical representation of information. They associate Data Components of the agent to the data inputs from these UI components, linking information of the Data Components of the agent to become a visual output.

For instance, for the graphic component (Figure 5.6) this process is demonstrated in Figure 5.7. The information requirements (defined in the GDTA) are linked to the Data Component of the agent, which are linked to the set of data input of the UI component, which are then rendered in run-time to the user.

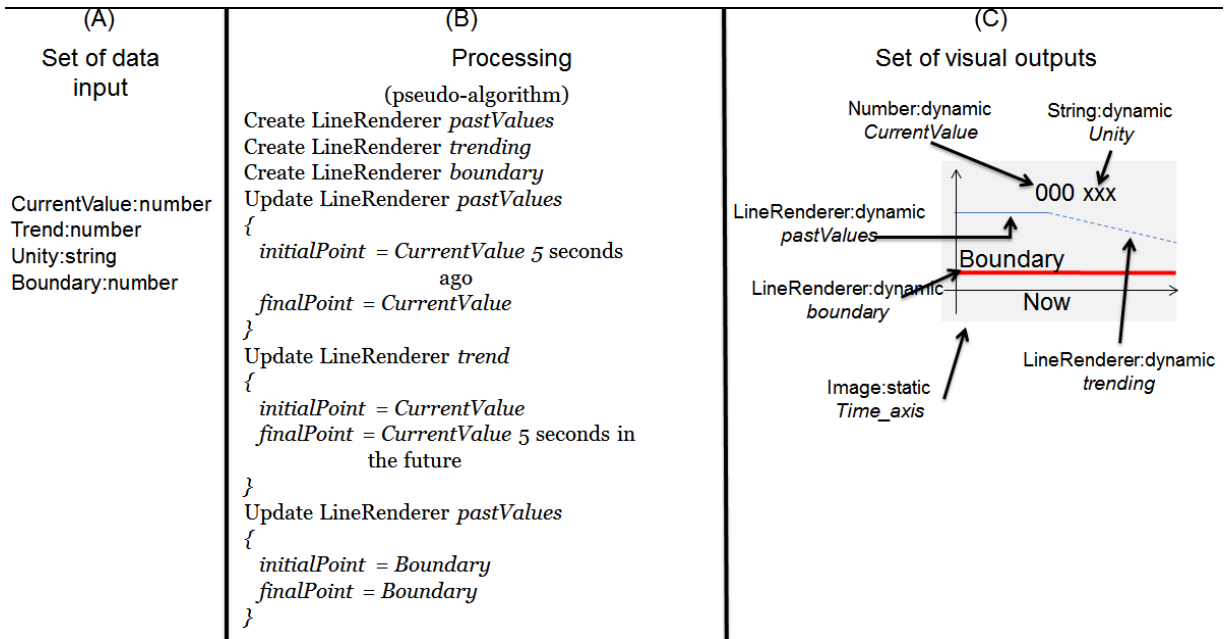


Figure 5.6: The creation of new UI components has three elements: (A) a set of data input, (B) a processing script and (C) a set of visual outputs using basic visual types.

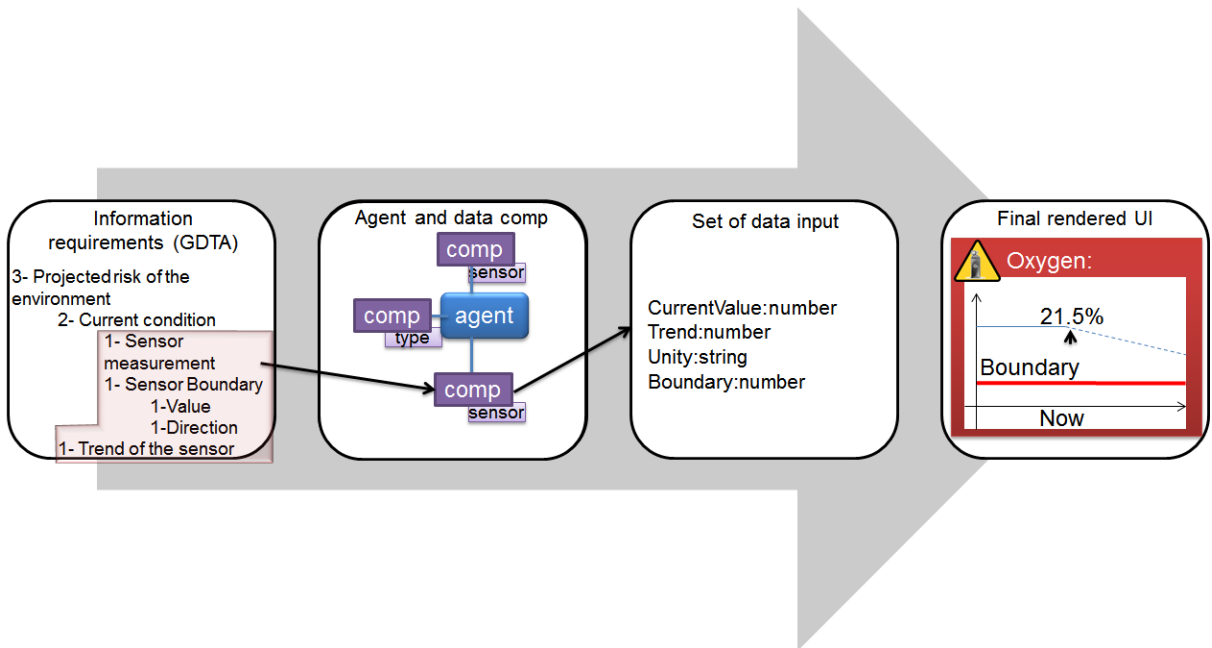


Figure 5.7: Process of linking Data Components of the agent to data inputs of the UI.

Metadata regarding each individual visual output, as well as default value as examples, can be defined to assist SMEs in understanding what is the purpose of each graphical element. Figure 5.8 shows two examples of metadata for two UI components, one that is a graphic of a value over time and the other that contains instructions for identifying broken equipment.

This step of the methodology is when UIs components that will use the resource of visual juxtaposition should be designed. This resource was explained in the SAOD guidelines, Section 2.3.2.2, as the guideline for supporting levels 2 and 3 of SAW.

Finally, due to the indirect manipulation approach of UI agents, user interaction is not treated by these components.

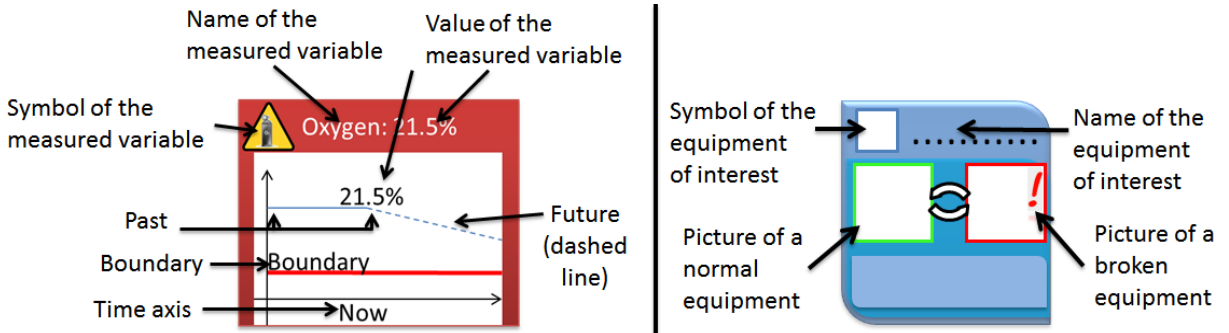


Figure 5.8: Example of metadata in UI components describing information fields.

5.4 Run-time architecture implementation

In this stage parts of the run-time will be implemented. The core of the architecture is already implemented through a framework, but two additions are important: Existing Agents and increments to the SA model. Figure 5.9 shows the process for implementing the run-time.

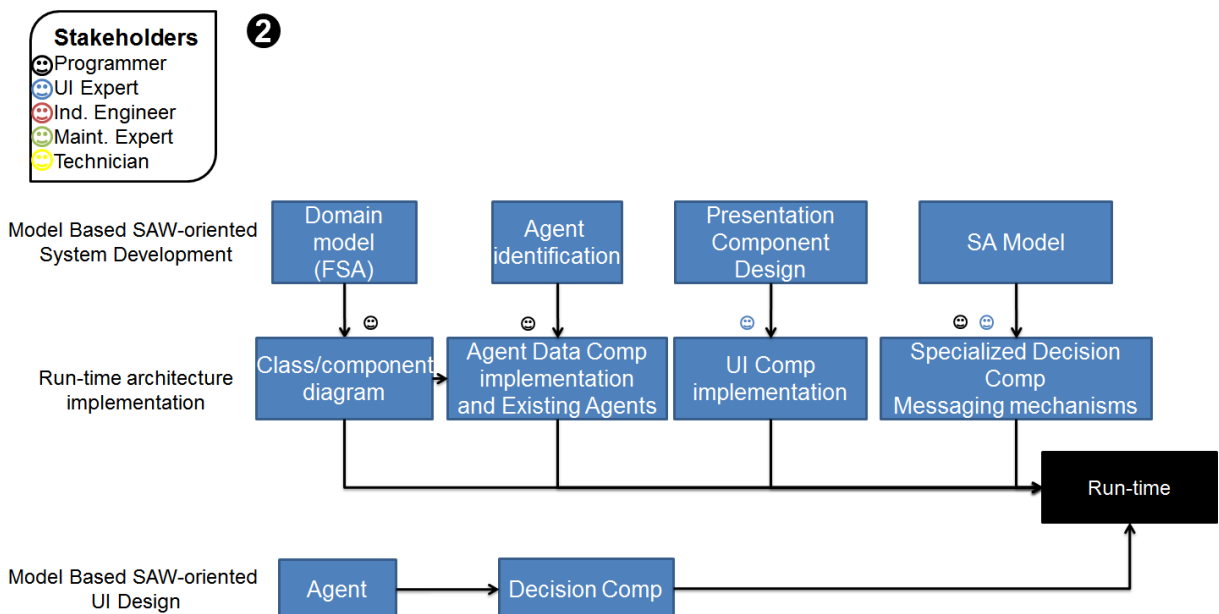


Figure 5.9: Run-time architecture implementation process.

From the FSA defined, a class or component diagram can be derived. Using for instance the FSA for the Environment Awareness [Oliveira15b], as expressed in Figure 5.4,

initially three components would be created: “sustainable development”, “plant layout”, “risks”. The properties of these components are the nodes connected to them.

After the component diagram is created, these components have to be implemented. Each component will become a Data Component to be used by agents. Data Components from the FSA presented in [Oliveira15b] were implemented for the study case (Chapter 6) and can be reused for most maintenance systems, but any alteration to the FSA demands implementation of new components.

These Data Components allow for easy access and usage of data of the agents during the UI Design stage. For instance, to access the noise sensor from the generator (an Environment agent), designers will create a script (in the Decision Component) that will be similar to: *Generator.Risks.Noise.Sensor.Value*.

This means that the agent Generator has a component of Risk, which has a component of the type Noise, which has a component of the type Sensor, which has a property of the name Value. In this property (Value) the value of the sensor is stored.

However, to really access the noise sensor value of the generator, an interface to this sensor needs to be implemented by the Data Component Sensor. That is why another step in implementing the run-time is implementing the Existing Agents.

These agents are the ones that need to access information from the E-Maintenance system, such as the generator just quoted. Their Data Components acts as wrappers for an interface programmed to access the E-Maintenance system with the information.

These interfaces are what distinct an Existing Agent. An Existing Agent needs these interfaces to access data, while some agents, such as some Procedure and Team agents, do not need to access any data directly from the E-Maintenance system, thus they can be completely created in the UI Design stage.

Continuing, in the run-time implementation stage, those additions determined in the last stage are also implemented. Therefore developers should implement now in the architecture any new UI components, new Specialized Decision Components, and new Messaging mechanisms.

With all implementation finished, finally SMEs and designers can create UIs, by creating agents and their decisions. Figure 5.9 shows all these steps, showing that in the final stage of the MBSAW-UI methodology (Model-Based SAW-oriented UI Design), agents and decisions are created to be instantiated in the run-time created in the second stage (run-time architecture implementation).

5.5 Model-Based SAW-oriented UI Design

In this stage, the base of the system is already implemented and all that it needs to be fully functional is defining task and their UIs. These tasks will be defined by industry experts (SMEs), which are the responsible for defining Work Instructions today in the industry, together with UI designers. Henceforth, to simplify, they will be addressed only as the designers.

Tasks are converted from current work procedures (or Work Instructions or even Internal Work Instructions) to the models defined in the Requirement Engineering stage.

Figure 5.10 gives an overview of the design process and its stakeholders. Each part of this process will be explained in the following sections.

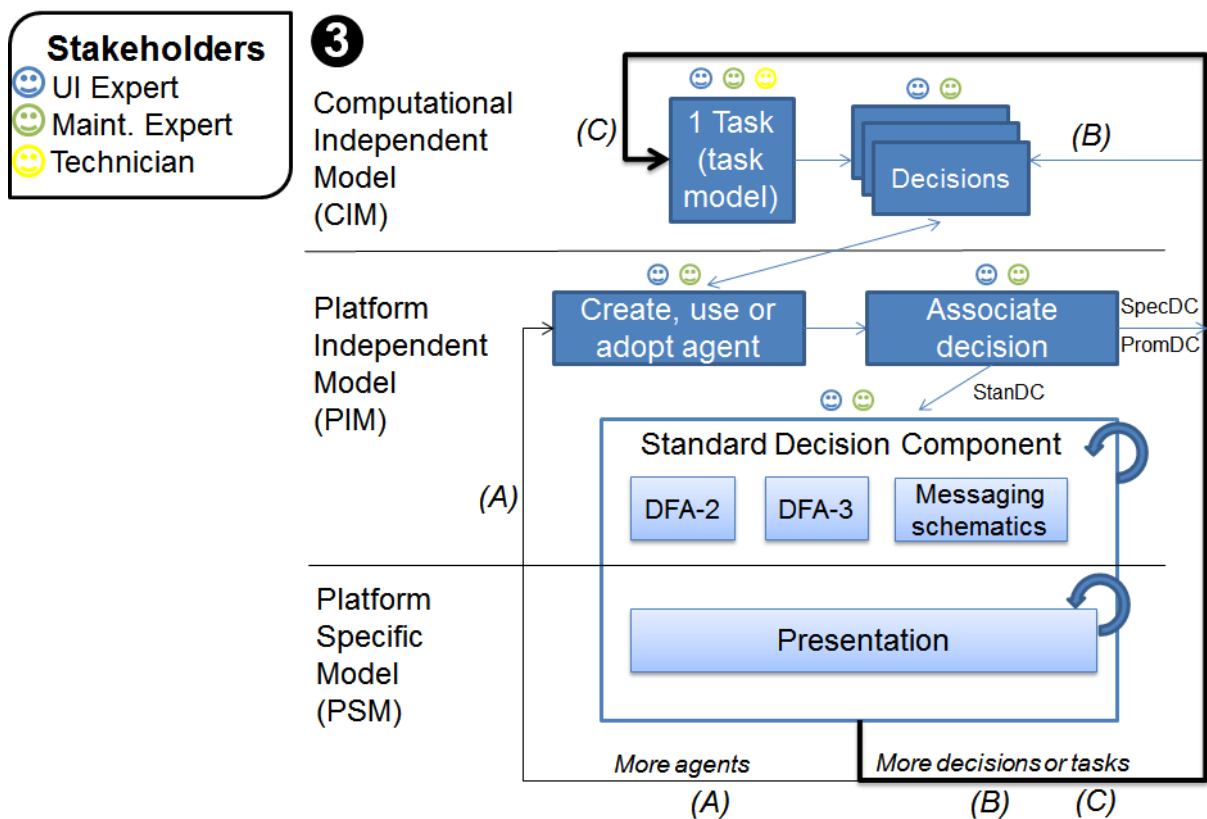


Figure 5.10: Model-Based SAW-oriented UI Design with stakeholders.

5.5.1 Tasks and decisions

The first step is to define tasks that will be modeled. Ideally each Work Instruction will become a task. Three examples of real tasks are: Inspection of the excitation system panels; Inspection and measurement of temperature of the output keys from generators;

maintenance of the compressed air system. With all tasks identified, one by one they are chosen to be modeled.

After a task is chosen, the first activity is to classify the task as one of the basic activities and do a Hierarchical Task Analysis (task model). Then, the GDTA from this specific task can be assembled using the decision-activity table.

As example the task of diagnosing a broken generator will be used. From the decision-activity table presented in Table 5.1 and the GDTA from Figure 5.3, a new GTDA for this task is made and shown in Figure 5.11.

Activities	Situation Aspects	Decisions
Common	Environment	What risks should be prevented?
	Equipment	Is it safe to start?
	Team	
Restoration	Equipment	What equipment should be isolated?
	Equipment	How to best fix the problems?
	Equipment	How to best reinstall the equipment?
	Equipment	Is it ready to be delivered?
Diagnostic	Equipment	What are the relevant problems?

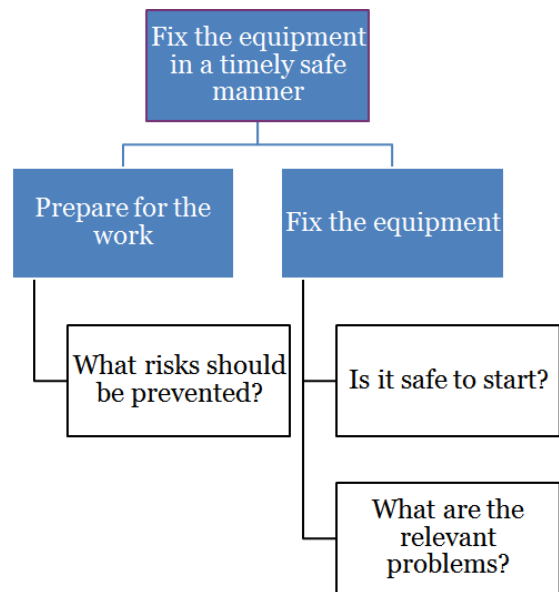


Figure 5.11: GDTA for the task of diagnosing a broken generator.

In the next step, now that the GDTA is ready, each decision will be modeled in the agents. To facilitate modeling Local SAW, a connection of tasks in the Hierarchical Task Analysis (HTA) to the decisions of the GDTA can be made. This way, in the next phase, which Local SAW will be the initial focus, it is easy to identify the decisions that need to be initially modeled (example in Section 6.4.2).

To move on to the next phase, one decision is chosen to be modeled. As observed in Figure 5.10, after this decision is modeled by an agent, this design process can continue in three different paths: (A) the decision has not been completely covered, and therefore more agents will model it; (B) there are more decisions to be modeled in this task, therefore another decision is chosen to be modeled; (C) all decisions of this task were modeled, therefore the process restarts with one other task being chosen to have its decisions modeled.

This cycle can go on as long as all the tasks are not covered, and even after, there is the possibility to update the system at any time, by changing a task or modeling a new one.

Continuing this process, the next section will debate how agents are modeled.

5.5.2 Decision modeling process

According to Figure 5.10, the decision modeling process is the aggregation of the second and third phases of the third stage of the MBSAW-UI methodology (PIM and PSM). But before going in detail about how each decision is modeled, it is important to cover the high-level process that guides this phase. Two terms should be defined beforehand: Agent Pool and Prompt Decision Component Pool.

The Agent Pool is the set of every agent in the system. Initially, it is only composed of the agents that were already implemented in stage 2, the Existing Agents. As new agents are now modeled from scratch, the Agent Pool starts to grow. As explained in Section 4.6.2, newly created agents become part of the Agent Pool and can be reused among different tasks.

The other term, Prompt Decision Component (PromDC), is basically a modeled Decision Component. After a decision is modeled in an agent, in some cases, this same decision will be applied to another similar agent. Since the StanDC is very flexible, remodeling the exact same decision in it can consume time, and therefore, the PromDC is a StanDC that is ready to be used.

For instance, a decision modeling risks (e.g. What risks should be prevented?) may be the same for many different environment agents. Of course the environments are different, but the decision-making of what and how risks should be prevented maybe the same.

Remodeling the same exact decision over and over should not be necessary, and the PromDC allows a decision to be reused in different agents. Since most decisions use dynamic data, obtained in run-time, as long as the agents have the same variables (the same Data Components) a decision can be reused.

In the decision modeling process, first agents are created, used or adopted. Adopting the agent means that an agent from the Agent Pool has already taken care of this decision, and therefore the process will immediately go on to the next decision. An example is when several tasks are already modeled, a new task to be modeled could adopt an agent (modeled in these past tasks) for its decision. That is why in Figure 5.10, the arrow going from “Decisions” to “Create, use or adopt agent” is double-headed.

Using an agent means using an agent from the Agent Pool to attribute the decision to, and creating literally means creating a new agent.

If the agent is created or used, the current decision is associated with this agent, and then it needs to be modeled. However, as made clear in Figure 5.12, there are three alternatives now: use the StanDC to model from scratch this decision; use a PromDC, a

Decision Component that has already been modeled; use the SpecDC to have an advanced model for this decision.

If StanDC or SpecDC are chosen, then the decision they will now model will also be included in the PromDC Pool, to be reused next time. Also, whatever alternative is chosen, this agent is now inserted in the Agent Pool (if it was not already as an Existing Agent).

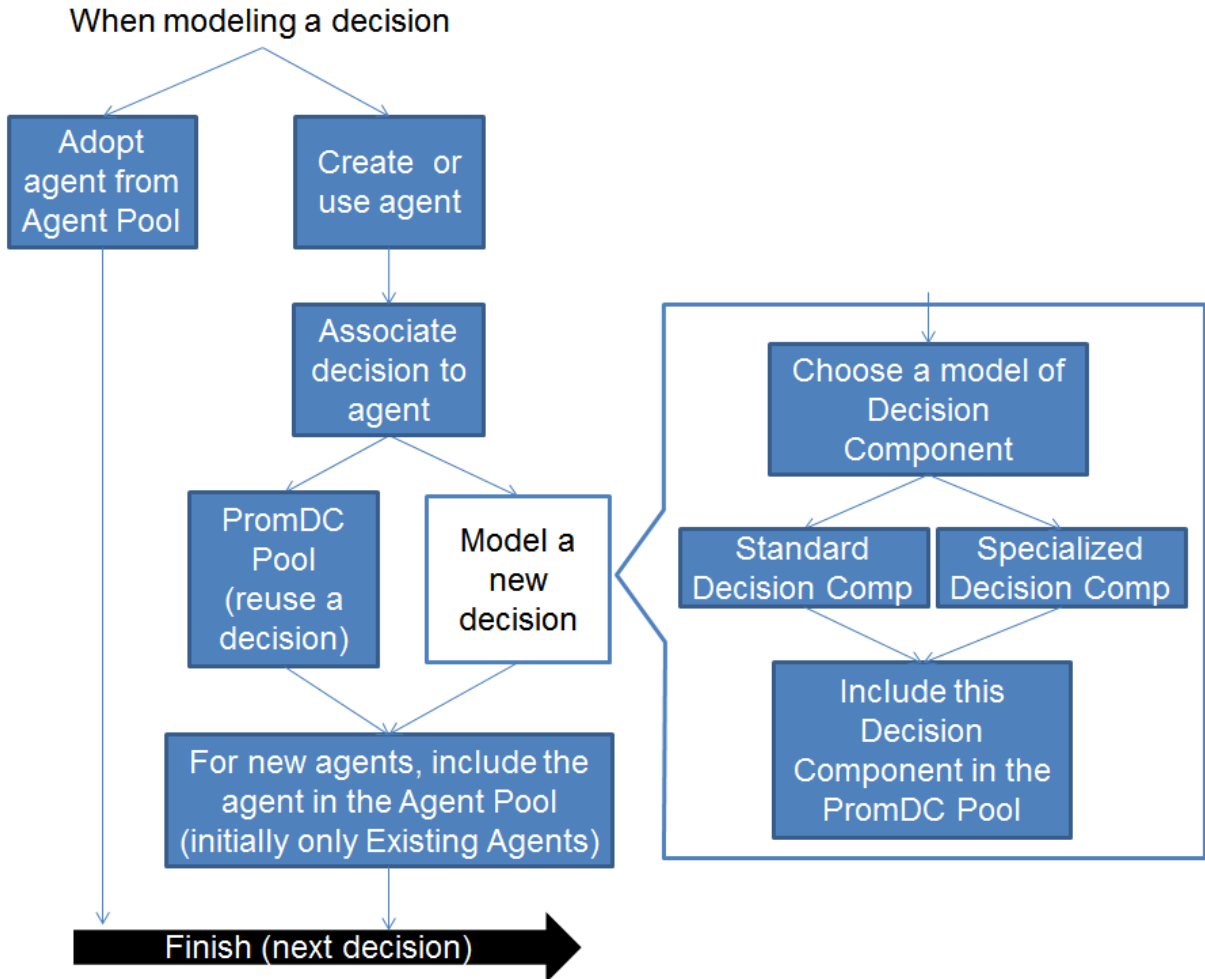


Figure 5.12: Overview of the decision modeling processing.

As an example, imagine an agent to represent an environment, named Room1. It will be initially created to model **one** decision, such as “Is there any danger in the environment?”. Noise is initially identified as a danger in Room1, and thus a StanDC will be used to create a Decision Component modeling noise as a danger. Then, height is also identified as a danger and subsequently modeled. By this point, two Decision Components are already attributed to the same agent (Room1), because even though it is the same decision (danger in the environment), noise and height are different problems, thus variants, for this decision.

Also by this point, Room1 is an agent in the Agent Pool, and the Decision Component for noise and height are both PromDC.

Later on, modifications are made in the Room1 environment (in the real world) and new dangers may be possible (electricity, heat, cold, oxygen, etc). Thus, there is the possibility at any time to define new Decision Components for Room1.

Additionally different decisions may also be modeled by this same agent, such as “where should collective protective equipment be deployed?” and “does the place needs cleaning?”. In the end, many decisions will be modeled for the Room1 agent.

The same dangers of noise and height, are also present in Room2. Therefore, for the same decision “Is there any danger in the environment?”, the PromDC of noise and height created for Room1 can be reused in Room2.

This high-level process gives a lot of flexibility and agility to the UI Design executed by the designers (SMEs and UI Designers). Since they will have to design every single UI for every single tasks, and normally there are hundreds of tasks, this process was devised to start with a focus on modeling new agents and decisions from scratch, but quickly develop into assigning/associating agents from the Agent Pool and PromDCs to new decision/tasks.

More examples of using the Agent Pool and PromDCs are given in the study case, in Chapter 6. The next section will explore the Standard Decision Component.

5.5.3 Standard Decision Component (StanDC)

According to Figure 5.10, after a decision is chosen to be modeled and an agent is associated with this decision, the process of modeling the Decision Component starts.

The Standard Decision Component structure is based in DFAs defined for levels 2 and 3 of SAW of the agent and the decision. Figure 5.13 shows how this structure works.

Every Decision Components (even those not using the StanDC) have a UI as output. In StanDC, UIs are defined for each state of each DFA and Messaging schematics are defined for each transition between states.

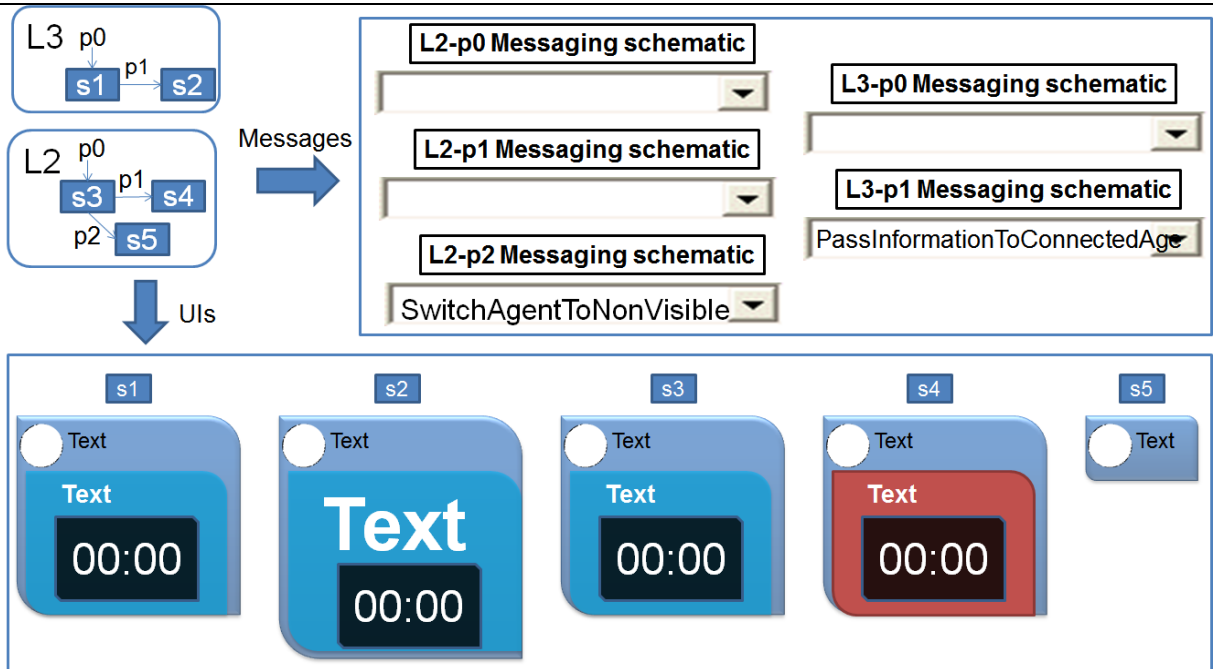


Figure 5.13: A modeled Standard Decision Component, with Messaging schematics defined for each transition, and UIs defined for each state of each DFA.

For instance in Figure 5.13, the DFA for level 3 SAW has two states (hence two UIs) and the DFA for level 2 has three states (hence three UIs).

The UI of the Decision Component at each time is calculated upon a Decision State Graph, which is the graph of every DFA from the Decision Component. Since there is always a superposition of UIs (since each DFA has a UI), the final UI uses a transformation scheme in which, at each state transition, it is transformed based on the current UI. Usually, the initial state of each DFA has an equal UI (s1 and s3 in this case), so at each state transition the UI is transformed a little, and transitions can be combined to generate many different UIs.

Using Figure 5.13 as example, if the L3 DFA was in the s2 state, and the L2 DFA was in the s4 state, the generated UI would be Figure 5.14.

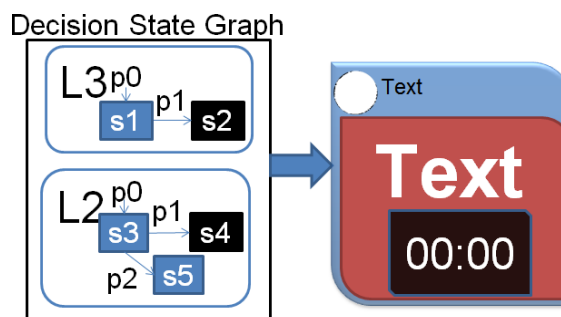


Figure 5.14: Decision State Graph and the final UI, based on each DFA state.

Transitions in the DFA also trigger the Messaging mechanism. For Figure 5.13, transition L2-p2 and L3-p1 have Messaging schematics defined.

To achieve these results, the process of modeling the StanDC is flexible enough to allow designers to approach their problems with multiple solutions. Figure 5.15 shows the overview of the steps necessary to model a StanDC and the next section describes them.

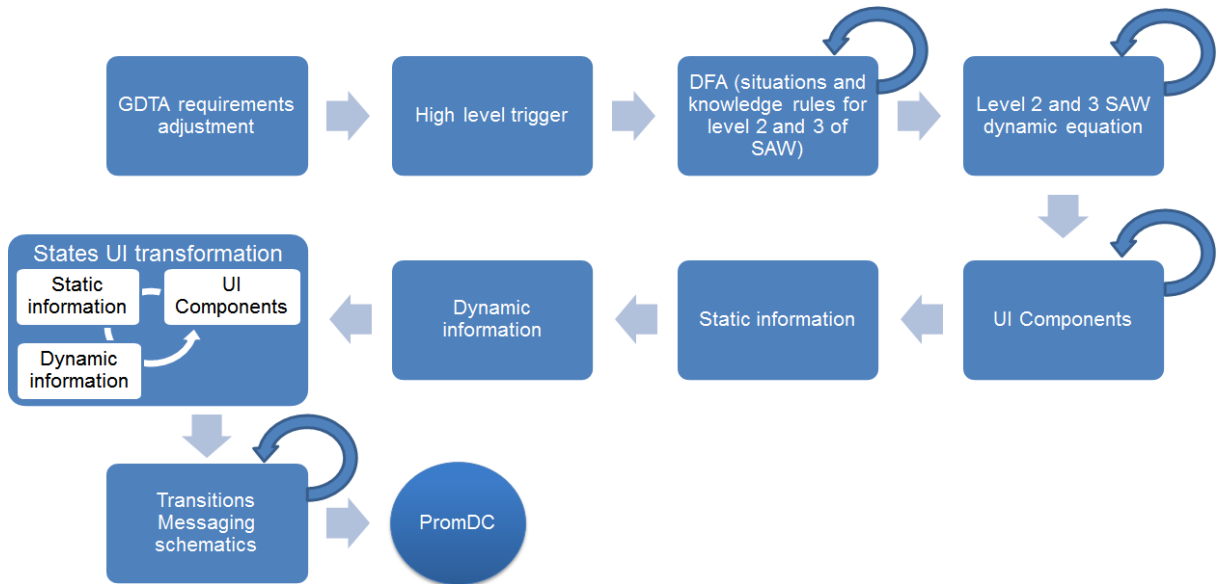


Figure 5.15: Overview of the process of modeling a Standard Decision Component.

5.5.3.1 GDTA requirements adjustment

In the first step of modeling a StanDC, designers need to define what are the information requirements for this decision. Information required is technically already established in the GDTA, but different equipment may have different information availability. Considering the decision “Which equipment should be inspected?” as an example, it has the information requirements of sensors values.

Nevertheless, not every equipment has a sensor, but they still need to be inspected. Therefore, in this step, designers can make adjustments to the information requirements defined in the GDTA, choosing what information is important. This is easy to execute **if** the GDTA created had as many information as possible (as recommended in Section 5.3.1.1).

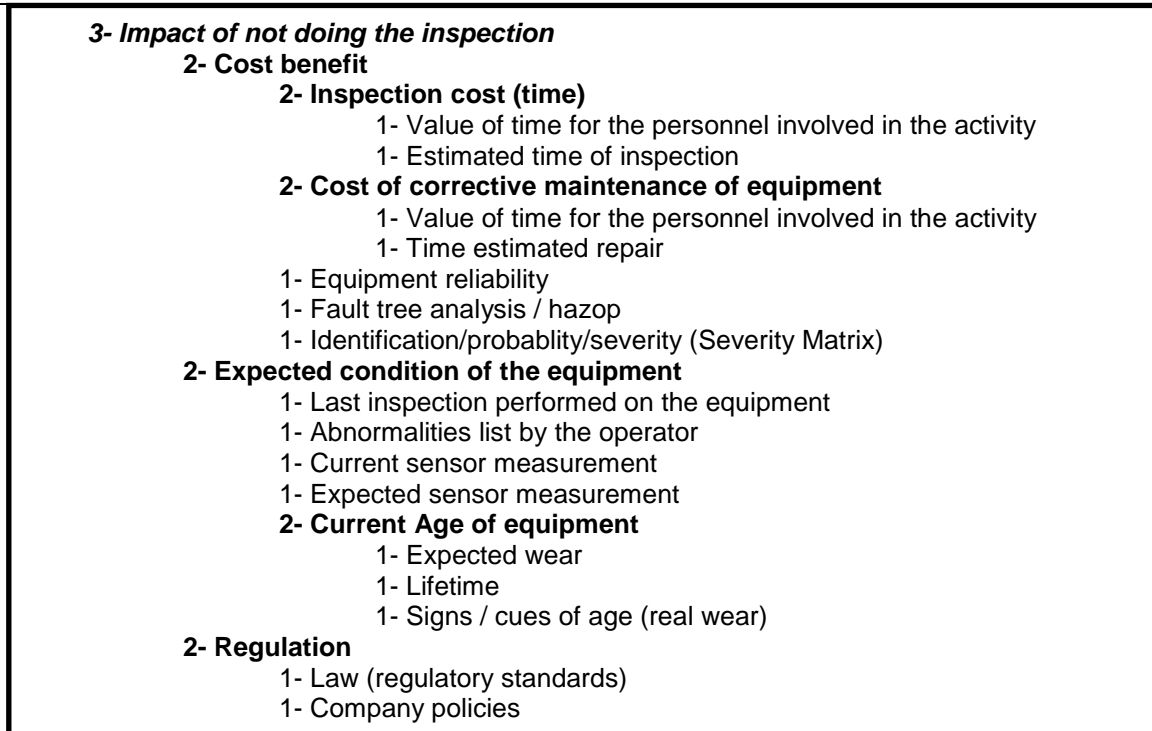


Figure 5.16: Complete information requirement of the decision “Which equipment should be inspected?”.

5.5.3.2 High-level trigger

In this step, designers appoint a high-level trigger required to start the decisions. These triggers can be:

- AreaEnter, AreaStay, AreaLeave, AreaTransition(A,B): a trigger that depends on location;
- StartTask, EndTask: a trigger activated when a task has started or ended. This is the most used trigger;
- StartSystem: a trigger activated when the system is initialized;

Before this trigger is activated, the Decision Component of the agent remains dormant. However, the agent is still awake, since agents can have several Decision Components, and they can still participate in the cooperation process (by supplying data requested by others).

5.5.3.3 DFA

The next step is defining the DFAs of the Decision Component. Each level 2 and 3 of SAW have their own DFA. So, if Figure 5.16 was used as example with its entire information requirement, the Decision Component would have five DFAs for representing the level 2 of SAW and one for a level 3 of SAW.

Nevertheless, designers can chose to not define a DFA for a specific level 2 or 3, which would means that the DFA has only one state. This is an important simplification resource, since it is not always important to define each level 2 situation. For instance, in Figure 5.16, designers may choose to not define a DFA for the “2- Current Age of equipment”, because the higher level information “2- Expected condition of the equipment” would already cover it.

An important consideration when creating these DFAs is that each state of the DFA is not a state of the entity. To make this distinction clear, an equipment agent (Generator) will be modeled with the decision “Which equipment should be inspected?”, the information requirements from Figure 5.16 and the DFA for the level 2 “2- Expected condition of the equipment”.

Each state of the DFA for this level 2 needs to represent a possible situation related to the “condition of the equipment”. These possible situations will later be used to define UIs to assist the user in comprehending the situation.

Transitions are defined via rules that use the Data Components of the agents. As an example, the DFA in Figure 5.17 was defined.

The initial state is named “Initial” and p0 is instantly activated when this Decision Component is triggered. p0 is always present and can be used to define a Message schematic. From the “Initial” state, transitions can occur to “Normal”, “Defect” and “Failure” situations. Each transition is explained below:

- p1: when the sensor of the equipment is below a boundary value, then the equipment is currently in a “Normal” situation;
- p2: if the equipment should be in the “Normal” situation (sensor ok), but the last operator report was too long ago, then the equipment the situation is “Inconclusive”;
- p3: if the operator report an abnormality not long ago, then the situation is “Defect”;
- p4: if the sensor value is higher than its boundary, then the situation is of a “Failure”.

A DFA for level 3 SAW has the same structure. For this example, the DFA for the level 3 “3- Impact of not doing the inspection”, would be the one in Figure 5.18.

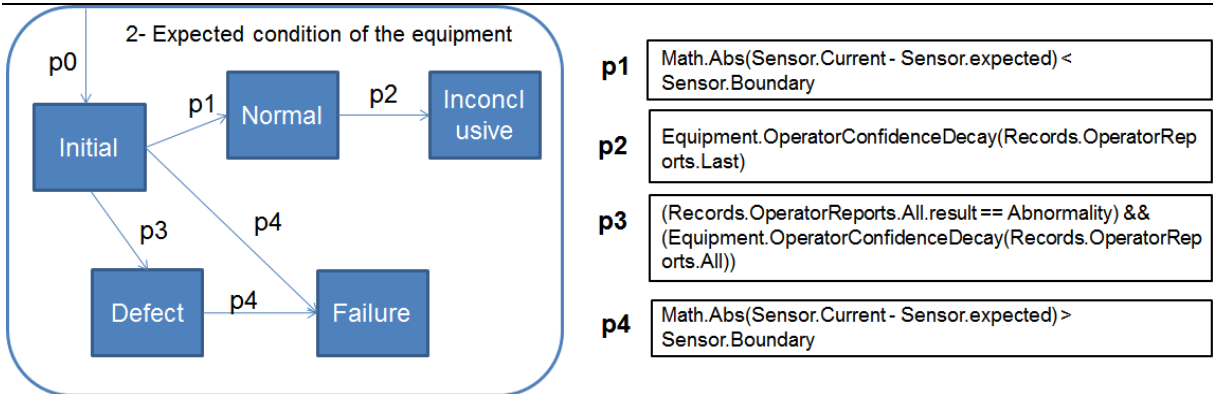


Figure 5.17: Level 2 SAW DFA with transitions for the decision “Which equipment should be inspected?”.

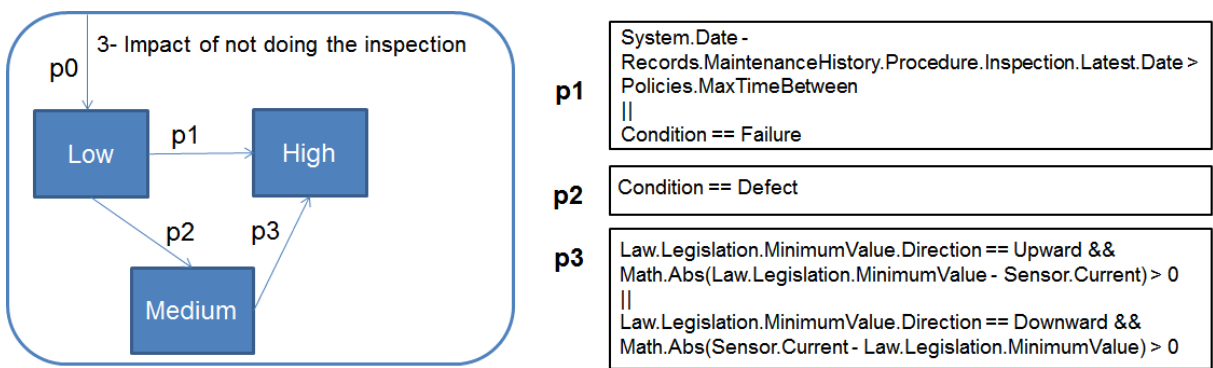


Figure 5.18: Level 3 SAW DFA with transitions for the decision “Which equipment should be inspected?”

Transitions for this level 3 DSA are:

- p1: when the last inspection was further than the maximum time allowed between the inspections, then the future situation (since it is a level 3 SAW, ergo projection) is “High” (impact), or if the equipment condition (situation, ergo level 2 SAW) is of “Failure”;
- p2: if the equipment condition is of “Defect”, then the impact is “Medium”;
- p3: if the equipment condition is of “Defect” and sensor is above or below the legal value, then the impact is “High”;

5.5.3.4 Levels 2 and 3 SAW dynamic equation

After DFAs are created, the next step is creating equations to express values of level 2 or 3 SAW. These equations are useful to directly express information in the UI.

For instance, when modeling a decision to avoid risk by time, the user has to be aware of how much time left he/she has. To calculate this time and show it directly to users, the following equation was used:

$$\text{Ent.Legislation.NR15.PermanencyTime}(\text{Env.current}) - \text{Env.Current.TimeInsideArea}()$$

This equation subtracts the time the user can stay in the environment (according to the Regulatory Norm 15 from Brazilian laws, which will be explored in the study case in Chapter 6) to the time he/she has already been there, showing only the time left.

Equations are created using the same structure of the DFA transitions, the Data Components of the agents. Combining this feature with the one for creating new presentation components (Section 5.3.3), it is possible to design any type of innovative UI for the system.

5.5.3.5 UI Components

The next step is defining the UI components for the Decision Component. The components that can be used are the basic visual types (image, string, number and line renderers) and any other component created for the system during the Presentation component design phase (Section 5.3.3). These components can be added one by one, until designers are satisfied.

Besides the visual aspect, designers also define if information in the components is static or dynamic. Static information will be set up by designers. Dynamic information will be obtained in run-time in the E-Maintenance system. Metadata to describe any information can also be added.

The final result from this step is a set of presentation components, with information marked as static or dynamic, like the example from Figure 5.19.

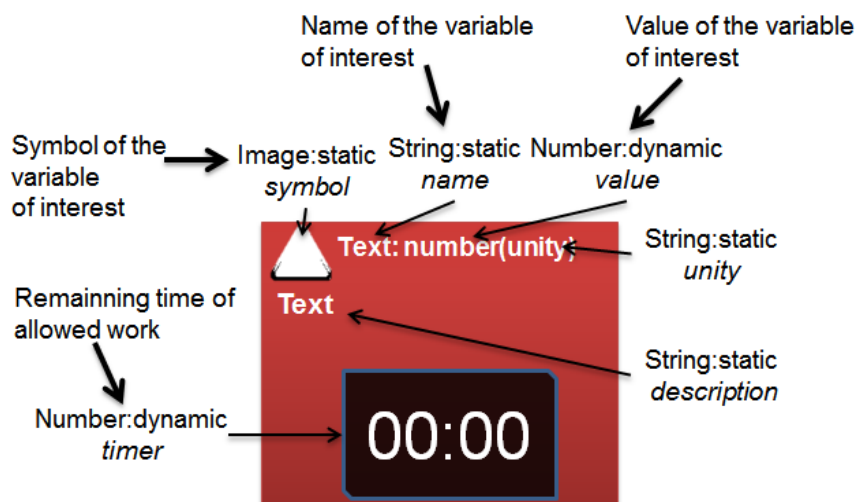


Figure 5.19: Defining the UI for the Decision Component, using only basic visual types and incremented Presentation components.

5.5.3.6 Static information

After the UI components are established, static information is defined. An example is Figure 5.20, which continues Figure 5.19.

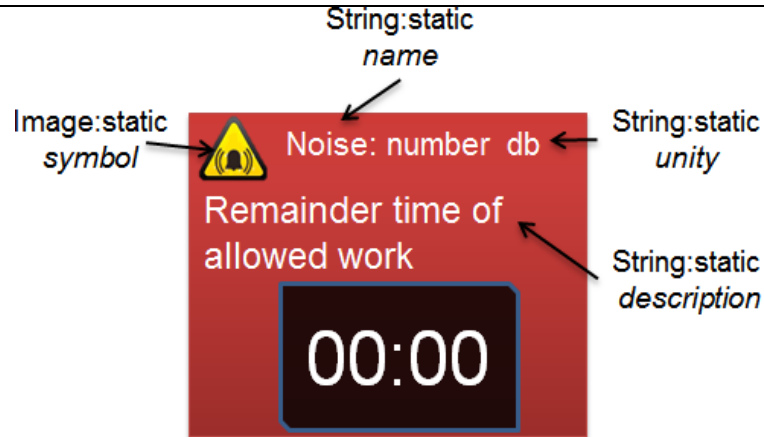


Figure 5.20: Defining static information in a UI.

This information is then saved on to the UI Component of the agent modeling the decision.

5.5.3.7 Dynamic information

In this step, as was already demonstrated in Figure 5.7, designers will link information from the Data Components of the agents to the dynamic information of the UI components.

Figure 5.21 continues the current example, showing the two dynamic information (value and timer) from the UI component being linked and the final expected result.

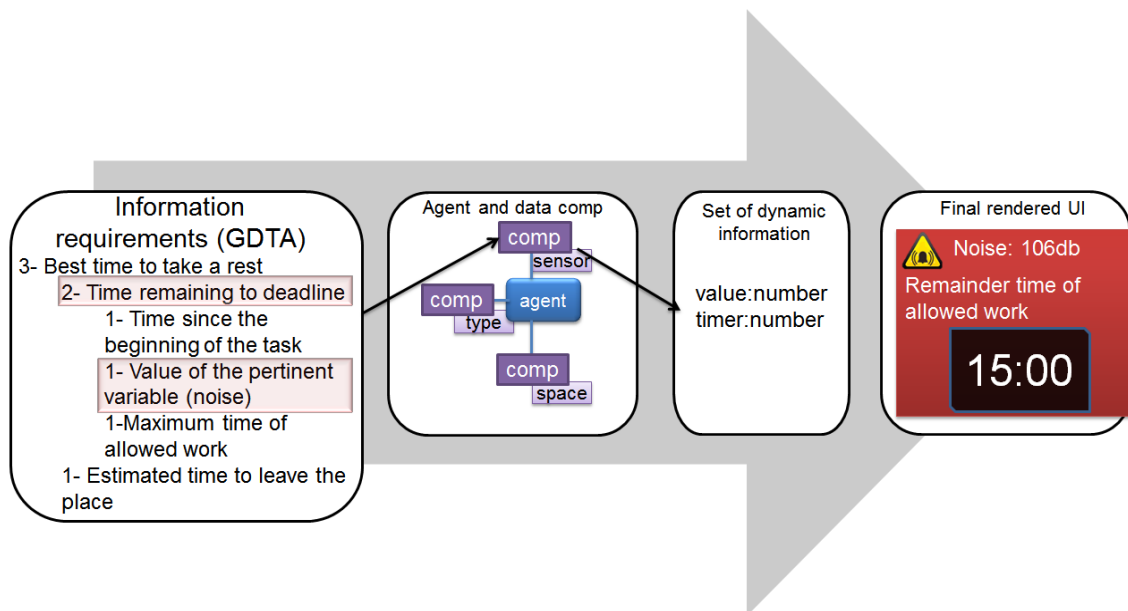


Figure 5.21: Process of linking Data Components of the agent to the dynamic information of the UI.

5.5.3.8 States UI transformation

The next step is defining UIs for each state of each DFA. As evidenced in Figure 5.15, this step is similar to the last three ones, but it is executed for each DFA.

Figure 5.13 already showed how this process works: for each DFA, designers define either variations of the UI or new UI components (or remove UI components).

The decision of changing, incrementing or removing is in the hand of the designers.

5.5.3.9 Transitions Messaging schematics

Finally, and also already explained in Figure 5.13, for each state transition of DFAs, Messaging schematics can be defined.

Table 5.3 shows examples of Messaging schematics, with the mechanism, target and additional properties of each schematic.

Table 5.3: Examples of Messaging schematics, with the mechanism, target and additional properties of each schematic.

Mechanism	Target	Properties
Alarm		
Increase salience	Procedure.Interruption.Source	
Create information bubble	Procedure.Current	
Switch procedure		Proc.EmergencyEvacuation
Pass information to agent	Procedure.Current.Target	“Recommendation: inspect”
Switch agent to Nonvisible mode	Equipment.Current.Design.Procedure	

5.5.3.10 Prompt Decision Component (PromDC)

After this Decision Component is finished, it becomes a Prompt Decision Component. This means that it becomes a Decision Component that can be used in other agents. The differences when reusing a PromDC in other agents, is that static data have to be redefined and dynamic data relinked.

Both examples given so far, one from Figure 5.16 to Figure 5.18, and another from Figure 5.19 to Figure 5.21, are good examples of PromDCs. The first example is modeling the decision “Which equipment should be inspected?”, and this decision needs to be modeled

inside several agents, with few differences among them. In the study case presented in Chapter 6, four Decision Components to model this decision were created, with one reused (as PromDC) seven times, two other reused two times, and only one was not reused.

Considering the decision “What are the possible risks to be aware?” (Figure 5.19 to Figure 5.21), the Decision Component shown was reused two times.

Therefore this feature is essential, because it saves a lot of rework. To close this chapter, the Specialized Decision Component will be addressed.

5.6 Specialized Decision Component (SpecDC)

A Standard Decision Component is flexible enough to model any type of decision the user will make. However, flexibility can damage cost-efficiency, because it may take too much time to model each Decision Component for every agent, and there may be hundreds of agents in a system. That is why there is a necessity for a Specialized Decision Component (SpecDC), speeding up the modeling in specific cases.

These specializations describe a decision model adequate for some specific situation scenarios. These will allow designers to quickly model several agents for these scenarios, but with diminished flexibility.

StanDC is as open and abstract as possible, to fit in any necessary case. On the other hand, the SpecDCs have to be instances of cases defined for an easier and faster design. For this thesis, five SpecDCs were made and are presented in Appendix B.

5.7 Final Considerations

This chapter presented the methodology MBSAW-UI, created to guide developers in designing and developing SASUIs for maintenance work. The methodology first proposes a Requirement Engineering stage, which guides in analyzing and defining the future system.

Then the next stage is the implementation of the architecture, as proposed in Chapter 4. The base of the architecture is already implemented, but in this stage increments are implemented according to requirements for the specific system in mind.

Then, in the final stage, designers (a combination of SMEs and UI designers) will model the User Interface by modeling the agents. In this process, they input every Work Instructions of their industry, making the system able to support any tasks and any decision by supporting the acquisition and maintenance of SAW by its users.

This strategy of using agents gives designers less control than traditional MBDUI approaches. On the other hand, they allow for a focus in the SAW (which is the goal). Agents will give this robustness while at the same time hiding major implementation details. Traditional MBDUI maybe overcomplicated for these SMEs and may also be too flexible, which is not necessarily required to design a SASUI.

Section 3.5 discussed how many UIs, design processes or systems, including some that used the model-based approach, do not follow basic guidelines of SAW, and are lacking in providing SASUIs. This methodology was made to fulfil this gap, giving developers and industries that want to create SASUIs for maintenance and industrial works a vision of how to do it.

To show how to apply this methodology in a real case, the next chapter presents a study case of an inspection of a generator in a hydro power plant.

Chapter 6

STUDY CASE: INSPECTION OF A HYDRO POWER PLANT GENERATOR

6.1 Initial considerations

A study case was implemented to both validate MBSAW-UI and its model-based run-time environment architecture, and to compare it to the default solution found today in most industries: a UI that supports primarily (and most cases only) Local SAW.

This study case was implemented first as a simulation independent of UIs. Thus, there is a procedure to be executed that can be done even without any supporting UI, just by having expertise of the task, and using the resources of the simulation.

The study case was implemented as a computer simulation after negotiation with experts from a hydro powerplant. Implementing the UIs for a physical task would have many downfalls: the correct visualization equipment (HMD) would be expensive and it does not fit with current safety gear; the real physical task has many dangers and implementing an UI for it before simulating and proving its validity would be irresponsible; permission to use the UI in a real task would be hard to acquire from company owners, to release us from responsibility in case of accidents; the task is only done every three to six months; the environment is dangerous and we would not be allowed inside to follow and observe the procedure

To implement this simulation, a technical visit was made to a site (a hydro power plant), data was gathered on the architecture and equipment of the power plant, and 3D models were created as a virtual representation of the plant.

To create a 3D model of the generator, a series of photos were used. The most important are shown in Figure 6.1: (A) a photo of a mockup of the generator bulb – side view; (B) a photo of the generator bulb from inside – front view; (C) a photo of the generator cover, which is above the generator bulb.

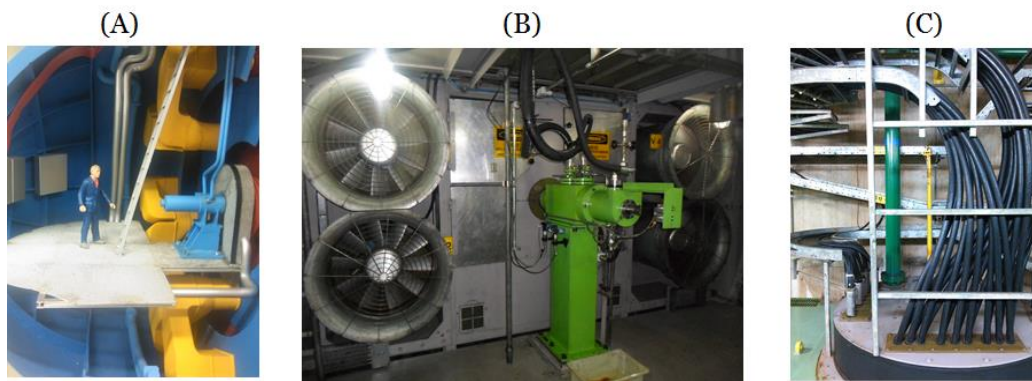


Figure 6.1: Photos of the real environment of a hydropower plant generator. (A) a photo of a mockup of the generator bulb – side view; (B) a photo of the generator bulb from inside – front view; (C) a photo of the generator cover.

These photos then led to the 3D models shown in Figure 6.2: (A) 3D model of the generator bulb – side view; (B) 3D model of the generator bulb – front view; (C) 3D model of the generator cover; (D) 3D model of the generator.

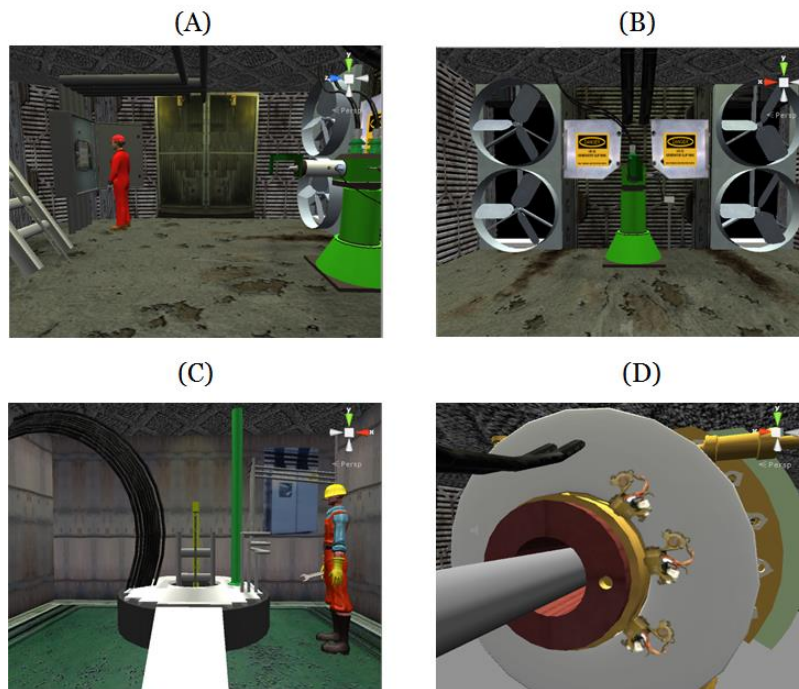


Figure 6.2: 3D models constructed to represent the power plant generator. (A) 3D model of the generator bulb – side view; (B) 3D model of the generator bulb – front view; (C) 3D model of the generator cover; (D) 3D model of the generator.

Then, a game development engine (Unity3D) was used to implement this simulation virtual world. Users can interact with the simulation through any computer. A version of the simulation using Oculus Rift (for Virtual Reality) and the Leap Motion (for gesture recognition) is under development.

After the simulation was implemented, tested and validated by SMEs, two UIs were designed and developed to assist users during the simulation.

The first UI was made to mimic most solutions in industry today. It was designed based on a HTA and therefore it gives a major assistance to users in developing Local SAW, but it does not offer any support for developing Global SAW.

The second UI was made using MBSAW-UI and its architecture, to support all the components of SAW that are necessary (Local/Global SAW and the three levels of SAW).

These UIs were then compared, by user tests, with both experts and non-experts in the domain. To facilitate discussion, the first UI was named Guide-UI and the second was named SAW-UI.

This chapter will focus on presenting the specification of the study case in details, and the development process of both UIs.

6.2 Description of the simulated task: inspection of a hydro power plant generator

6.2.1 Task, domain and cognitive analysis

The task simulated is an inspection of a generator in a hydro power plant. Since this is an inspection, the majority of work is executed (and evaluated) through reporting of problems.

This was based on a real task, which is executed approximately every 3 to 6 months in any power plant. However, some of the procedures of the Work Instructions that defines this task were left out, to make the task a little shorter than it is in real life.

A task analysis appointed eight basic procedures for this task:

- Procedure 1 – Gather Personal Protective Equipment (PPE): users start by equipping a latch belt, helmet, earbud, boots and anti-flame outfit. They proceed to get a series of tools, which are a lantern, radio, thermometer and gas measurer. The lantern need to be tested, and if it is not working, users should report this problem and acquire (and test) another lantern.
- Procedure 2 - Inspect generator cover: users need to deploy 4 cones (Collective Protective Equipment) around the stairs connecting the generator cover to the bulb, and then check for cleaning problems and defective lamps.
- Procedure 3 - Enter generator: users need to attach the latch belt and then climb down the stairs.

- Procedure 4 - Inspect bulb: users check for cleaning problems, defective lamps and fans, and abnormal noises.
- Procedure 5 - Inspect Kaplan cylinder (rotor) head: users verify vibration and oil leakage of the cylinder head, and then check three PT100s (a temperature sensor) for broken wires and high temperatures.
- Procedure 6 - Inspect Slip Ring: users investigate the slip ring for overheating, vibration, and sparks. They then proceed to check feeding cables A and B for overheating.
- Procedure 7 - Inspect brush holder: users inspect three brush holders for wear of brushes and misalignments, and then their tails (cooper wire) for overheating and broken wire.
- Procedure 8 - Exit and classify the maintenance: users need to attach the latch belt and then climb up the stairs. To conclude the task, they must analyze the problems reported and classify the type of maintenance recommended for the problems.

From this list, the Hierarchical Task Analysis (HTA) from Figure 6.3 was defined.

In each procedure, users could report problems/defects spotted. They were also given the option to skip any procedure (or a step in a procedure) if they judged it unnecessary.

The problems/defects were called Abnormalities in Operative Conditions (AOC) and a total of 24 were implemented, which can be turned on or off to generate a variety of situations in the simulation. Besides equipment defects, AOCs can also be incidents prone to happen (e.g. fire, gas leak and oxygen drop).

At the end of their task, based on the problems reported, users should decide on four recommended maintenance actions: nothing, programmed, corrective with urgency or emergency.

A domain analysis revealed the simulation should also accounts for safety legislations, based on the Regulatory Norms (RN) from Brazilian laws, which dictates safety measure for workers (similar to OSHA). They go from using protective equipment, to using a fall arrest when going up and down the stairs, to a limited time of exposition to heat and noise.

Four RNs formed the basis of safety in the simulation: RN-5 (internal commission to prevent accidents), RN-10 (work with electricity), RN-15 (insalubrious environment) and RN-33 (confined space).

Noise and heat are part of RN-15, about activities in insalubrious environment. Since noise is 106dB in the simulated environment, users can only stand in this environment for 25 minutes before compromising their health. This value is even higher (110dB – 15 min) when the AOC “Abnormal Noise” is on. Heat is expressed as the Wet Bulb Globe Temperature

(WBGT) index, a widely used and accepted assessment of heat stress in industry, and it has a value of 31.5 when the AOC “Two Broken Fans” is on (this imply that the user needs to rest 45 minutes for every 15 of work).

Finally, a Cognitive Task Analysis was performed, in the form of a Goal-Directed Task Analysis (GDTA). The GDTA for the generator inspection was summarized in Figure 6.4 (first level are goals, second level are decisions). The complete GDTA for maintenance is presented in Appendix A

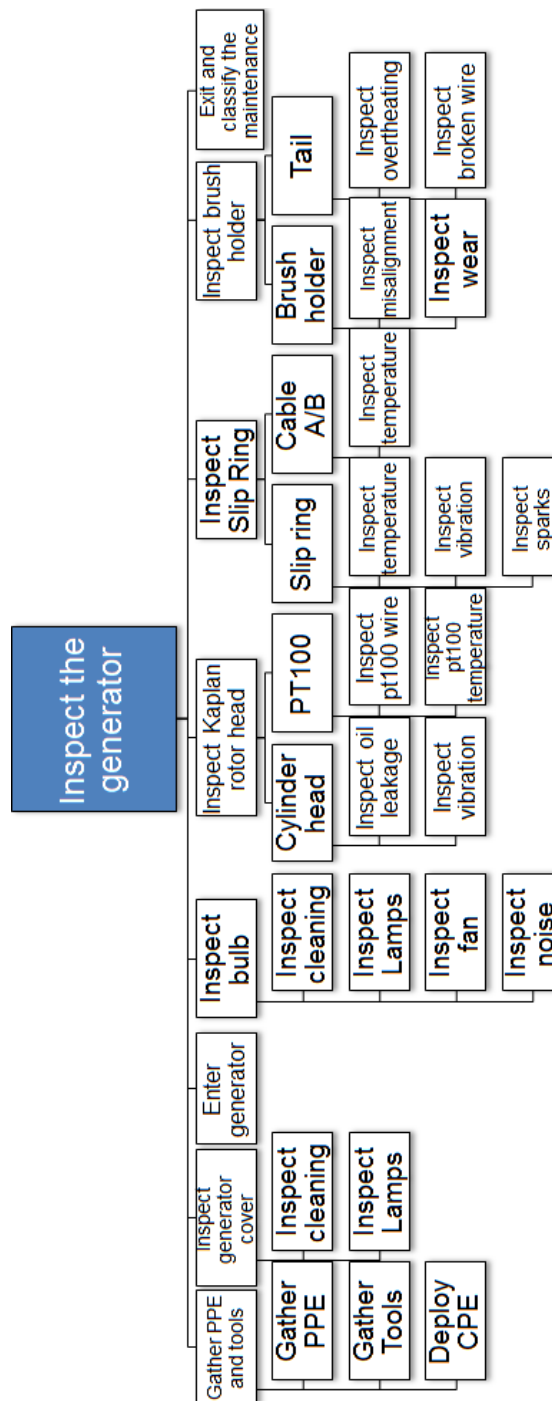


Figure 6.3: Hierarchical Task Analysis of a hydropower plant generator inspection.

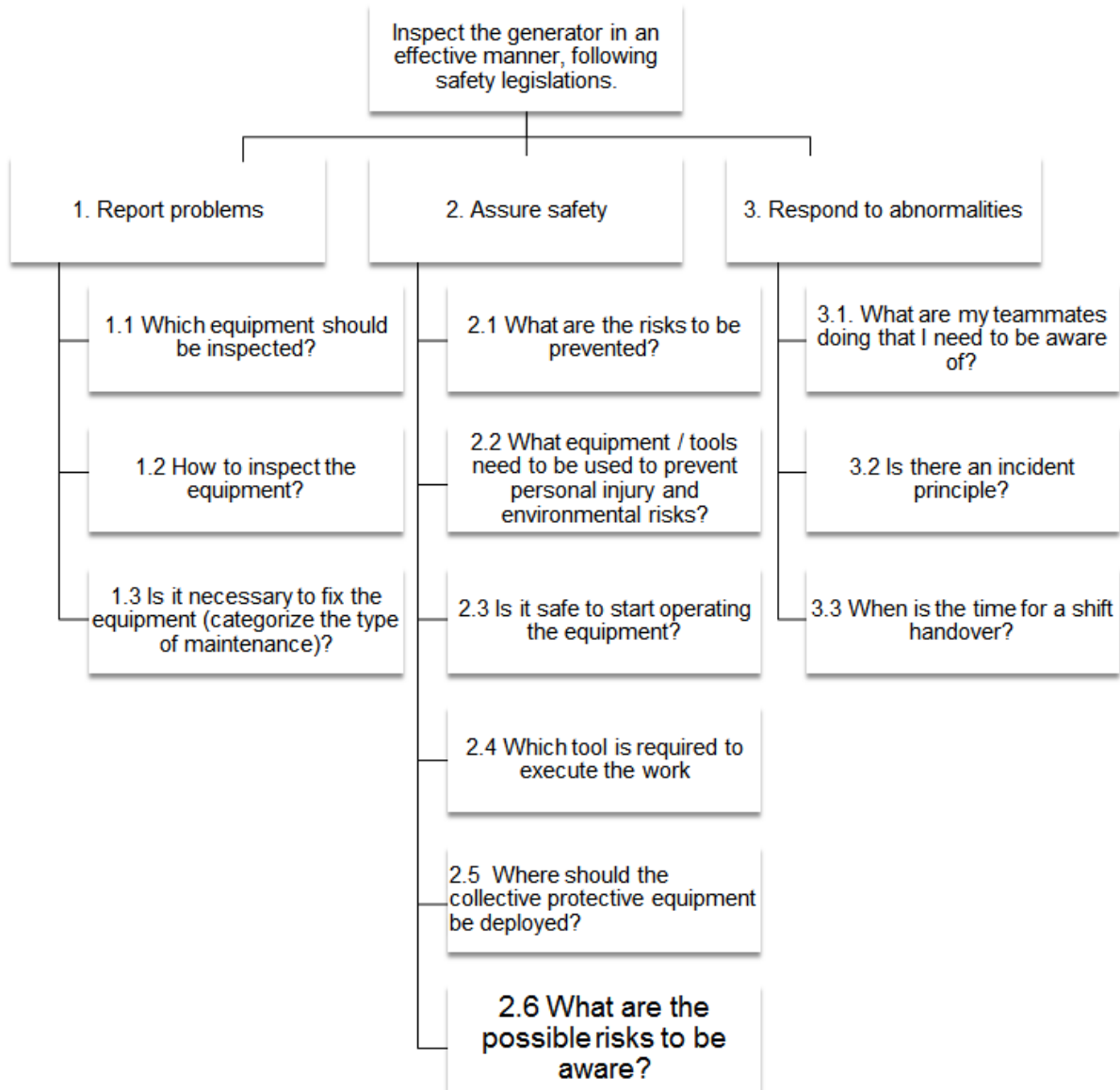


Figure 6.4: Goal-Directed Task Analysis (GDTA) of the task of inspecting a hydropower plant generator.

6.2.2 Simulation and situations

With the total of 24 AOCs, three different situations were assembled to randomize enough content so that users could test it several times (thus increasing data to analyze results) without repeating the problems. Table 6.1 shows the AOCs for the three situations designed.

Table 6.1: Three situations, each one with eight Abnormalities in Operative Conditions (AOC), were designed for the task of inspecting a hydropower plant generator.

S	Abnormalities in Operative Conditions (AOC)							
S 1	Cleaning problem in the cover	Oil leak in the cylinder head	Vibration in the cylinder head	All PT100 with broken wire	Vibration in the slip ring	Spark in the brush holder	Over-heat in the tail	Brush wear
S 2	Broken lamp in the cover	Cleaning problem in the bulb	2 broken fans	PT100 with high tempera- ture	Over-heat of slip ring	Broken wire (tail)	Brush holder misa- ligned	Fire
S 3	Broken lamp in the bulb	Lantern not working	Abnor- mal noise	1 broken fan	Cowor- ker interrup- tion	1 PT100 with broken wire	Over-heat cable	Gas leak

Each situation was assembled with an expectation. S1 is an introductory situation (without any major incident or deviation planned), focused on the cylinder head and slip ring. It is good to introduce the simulation to both newcomers (non-experts), with all major equipment of a generator, and experts, with the representation of the problems/defects.

In S2, WBGT has a value of 31.5 (because of the two broken fans). Therefore users have 15 minutes to work before they take a 45 minutes break, and then they can work 10 more minutes before taking a one-day break (because of the 106dB of noise, which allows a total of 25 daily minutes of work). In the end of S2 users are also presented to their first incident, a fire that starts as soon as they start procedure 7. In this case the correct response is immediate evacuation (due to the nature of the confined space).

In S3 users will be interrupted for 7 minutes by a coworker when passing to procedure 5 (usually around the 8-12 minute mark). In this case, since the noise is higher because of the AOC “Abnormal noise” (110dB, which means 15 daily minutes of work), the response expected from users is to anticipate the rest, instead of waiting for the interruption to stop before taking the rest. Finally, in the end of S3 a gas leak happens (dropping the oxygen), which makes a coworker faint. The correct response is immediate evacuation while carrying the coworker upstairs.

The simulation offer as resources options for users to report problems in equipment, since the task simulated is an inspection, and also offer options to countermeasure many problems presented, such as: being able to take rests when necessary; being able to finish the simulation at any time; being able to carry a coworker that passed out; being able to carry and deploy cones for collective protection; being able to acquire Personal Protective Equipment (PPE); being able to use a latch belt (one of the PPEs) to go down stairs; being able to read a temperature and oxygen measurement gear (tools); being able to use a lantern for better vision; being able to freely access any place of the simulation (the generator cover, the bulb, and the generator).

To simulate the answer to insalubrity problems (resting), users could come back to the starting point of the simulation at any time and take a one hour or one day stop (i.e. they would do something else during period). Figure 6.5 shows the starting point screen options and the data users were informed after starting a new simulation.

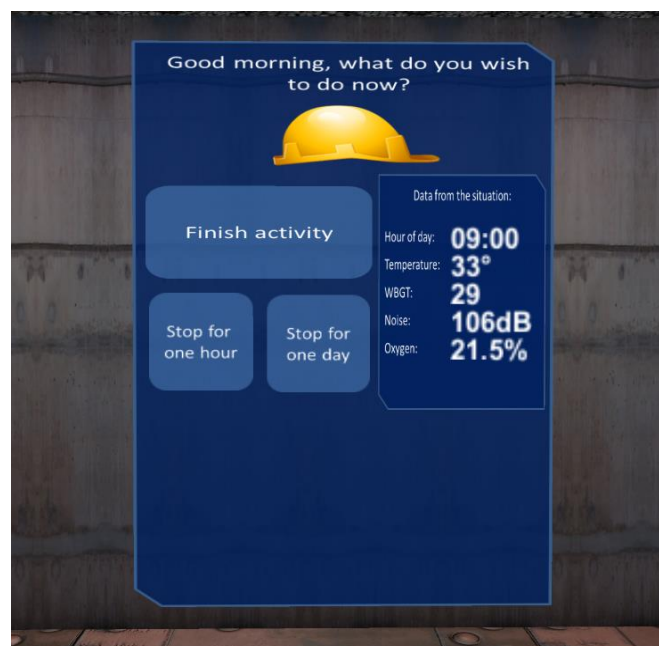


Figure 6.5: Simulation control interface, to allow users to rest during their task.

6.2.3 Efficacy and Situation Awareness analysis

An efficacy analysis pointed that there are 32 possible errors that users can commit. The majority of the errors are related to the procedure itself, such as reporting nonexistent problem, or not reporting a problem.

To complement this analysis, mistakes solely related to erroneous Situation Awareness were included. These mistakes are not considered errors in real life, however they

demonstrate a clear lack of SAW that a SASUI is expected to fix. One example happens in S3, as explained, during the coworker interruption the user ideal response is to anticipate the rest, and not doing it is considered an error (in this simulation).

With the analysis completed, each error was classified as a lack of awareness in one or more situation aspects of the FSA. Appendix C shows this list of errors, their corresponding lack of awareness, and how many times they were committed by users with each UI.

Procedure errors are about forgetting to do a procedure listed in the original work instructions. Equipment errors are about misinterpreting an equipment condition or mishandling PPE. Team errors are related to lack of coordination or assistance to coworkers. Enterprise errors are about not executing obligatory legal procedures (the Regulatory Norms - RNs). Environment errors are about misunderstanding environmental variables and conditions.

These errors were classified using the FSA so that later, the Local and Global SAW performance of users are analyzed. Errors could also belong to two types of Awareness (in theory even more, but they were restricted to the two most important). For instance, reporting nonexistent problem, or not reporting a problem, both are a lack of Procedure and Equipment awareness (either the user did not follow the procedure or did not understand the condition of the equipment).

Another example is not taking a rest when the noise or WBGT timer reaches its deadline, which is a lack of Environment and Enterprise awareness (either the user is not aware of this risk in the environment, or not following a company/government legislation - NR15).

One last example is not rescuing the coworker in S3, which is a lack of Team and Enterprise awareness (either users are not aware of their teammate condition, or not following a company/government legislation - NR10).

6.3 Guide-UI development: focus on task analysis

Guide-UI is a simple visual abstraction of the work instruction, a mobile UI based on paper documentation without any increment, to be compared with the SAW-driven solution. It was made using the HTA from Figure 6.3. Figure 6.6 shows Guide-UI for procedure 4.

As demonstrated in Figure 6.6, Guide-UI only shows a basic description of the activities. The UI also shows a clock and the thermometer and gas measurer.

This UI never warns users about timers (they can use the clock to measure their own timers) and it has only two alarms, which are activated as the sensors pass a boundary (oxygen and temperature), to simulate the thermometer and gas measurer.

Figure 6.7 shows all the UIs designed for Guide-UI, each one for their correspondent procedure (in the UIs the procedures are called activities).



Figure 6.6: Guide-UI for procedure 4 (inspecting the bulb).

<p><u>ACTIVITY 1</u> Gather PPEs:</p> <ul style="list-style-type: none"> • Radio • Lantern • Latch belt • Helmet • Boots • Earbud • Anti-flame outfit • Gas measurer • Thermometer 	<p><u>ACTIVITY 2</u> Inspect generator cover:</p> <ul style="list-style-type: none"> • Put cones around the cover for collective protection • Check cleaning conditions • Check the 3 lamps operation 	<p><u>ACTIVITY 3</u> Enter generator:</p> <p>⚠ Warnings: Use latch belt to access the generator</p>	<p><u>ACTIVITY 4</u> Inspect the bulb:</p> <ul style="list-style-type: none"> • Check the fans operations • Check for strange noises • Check cleaning condition • Check the 7 lamps operation
<p><u>ACTIVITY 5</u> Inspect rotor head:</p> <ul style="list-style-type: none"> • Check abnormal vibration • Check oil leakage • Check PT100s for: <ul style="list-style-type: none"> • broken wire • high temperature 	<p><u>ACTIVITY 6</u> Inspect slip ring:</p> <ul style="list-style-type: none"> • Check for sparks • Check for abnormal vibrations • Check for overheating • Check for overheating in feeding cables A and B 	<p><u>ACTIVITY 7</u> Inspect brush holder:</p> <ul style="list-style-type: none"> • Check brush holders for: <ul style="list-style-type: none"> • brush wear • misalignment • Check tails for: <ul style="list-style-type: none"> • broken wire • overheating 	<p><u>ACTIVITY 8</u> Exit generator:</p> <p>⚠ Warnings: Use latch belt to exit the generator</p>

Figure 6.7: Guide-UI and all the UIs for every procedure.

6.4 SAW-UI development: a Situation Awareness Support UI (SASUI) developed using the MBSAW-UI methodology

SAW-UI was designed using the MBSAW-UI process. In the next subsections the complete process will be presented.

6.4.1 Requirement Engineering

6.4.1.1 GDTA and FSA

The GDTA was already presented in Figure 6.4. The FSA is presented in [Oliveira14, Oliveira15b].

6.4.1.2 Activities

Beforehand, by analyzing several Work Instructions documents, five types of activities were defined, which compose all possible activities in maintenance. They are presented below, together with a few synonyms found in work instructions that indicate their probable classification:

- Concerning preventive maintenance:
 - Inspection: inspect, check, compare, verify, ascertain
 - Restoration: restore, change, clean, lubricate, discard, unblock, complete (oil), calibrate
 - Assembly: assemble, disassemble, align, install, isolate, loose
- Concerning corrective maintenance:
 - Diagnose: diagnose, analyze, measure, gauge, take readings
 - Restoration
 - Assembly
- Concerning predictive maintenance:
 - Test: test, turn on, turn off, use, rehearse (do test)

Not every maintenance type was covered (for instance opportunistic maintenance).

Then, the decision-activity table was established, defining what decisions belong to each activity, as shown in Table 6.2.

Table 6.2: Relation between activities and decisions.

Activities	Situation aspects	Decisions
Common	Env Equip	What are the risks to be prevented?
	Equip	Which equipment and tool should be used to prevent personal and environmental risks?
	Equip	Which tool is required to execute the work?
	Equip Team	Is it safe to start operating the equipment?
	Equip	Where should the collective protective equipment be deployed?
	Env Equip	What are the possible risks to be aware?
	Sys	What automations should be acknowledged during the task?
	Team	What are my teammates doing that I need to be aware of?
	Env Team	Is there an incident principle?
	Team	Who needs to be informed of the end of a task?
	Env	Does the workplace needs cleaning after the task?
	Proc	Does any difficulty during the work needs to be reported?
	Sys	What information are important to keep track?
	Inspection	Equip
Equip		How to inspect?
Equip		Is it necessary to fix the equipment now (classify the maintenance)?
Testing	Equip	Which equipment should be isolated to execute the task?
	Equip	Which testing strategies need to be used in the equipment?
	Equip	Is the equipment ready to be integrated to production again?
	Equip	Is it possible to remove isolation to reintegrate the equipment?

Activities	Situation aspects	Decisions
Assembly	Equip	Which equipment should be isolated to execute the task?
	Equip	Is the equipment ready to be integrated to production again?
	Equip	How to dis/assemble the equipment?
	Equip	Is it possible to remove isolation to reintegrate the equipment?
Restoration	Equip	Which equipment should be isolated to execute the task?
	Equip	Is there enough material (assets) to execute the task?
	Equip	How to solve the problem?
	Equip	How should the equipment be fixed?
	Equip	Is the equipment ready to be integrated to production again?
	Equip	Is it possible to remove isolation and reintegrate the equipment?
	Equip	How to discard removed spare parts?
Diagnostic	Equip	Which equipment should be isolated to execute the task?
	Equip	What is the equipment defect? (diagnosis of any failure/defect of the equipment)
	Equip	Is it necessary to fix the equipment now (classify the maintenance)?
	Equip	Is it possible to remove isolation and reintegrate the equipment?

For this study case, only inspection procedures will be approached. The GDTA from Figure 6.4 was created using this table.

6.4.1.3 Agent Identification

To identify agents, initially a table with each procedure (P1 to P8) was created, and each procedure was analyzed for real-world entities (agents). The results are compiled below:

- Procedure agents: one agent to model procedures P1 to P8, named as the Generator Inspection (GI) Procedure agent;

- Equipment agents: each equipment from the procedures is an agent. They are: PPE (Personal Protection Equipment), which are latch belt, helmet, earbud, boots and anti-flame outfit; tools, which are lantern, radio, thermometer and gas measurer; cones as Collective Protection Equipment (CPE); lamps, in the cover and the bulb; fans; CTG panel; cylinder head (with three PT100); slip ring, with three brush holders, each one with a tail (cooper wire); feeding cables A and B;
- Environment agents: seven environments are agents: the generator cover, the stairs, the bulb, the generator case, the starting point, the passage (connecting the starting point to the cover). The last environment, the generator, aggregates all of the previous ones inside it;
- Team agents: only one team agent was created, to represent the team working on the task. The team had three coworkers, the user, a scout (Rogers) and another working on the CTG panel (Kurt). Kurt and Rogers are bots in the simulation;
- Enterprise agents: two enterprises agents were created, the plant owner (company) and the Regulatory Norms owner.

Figure 6.8 shows a schematic of the environments (their disposition), equipment (the environment where they are), team members and procedure (where they are executed) agents.

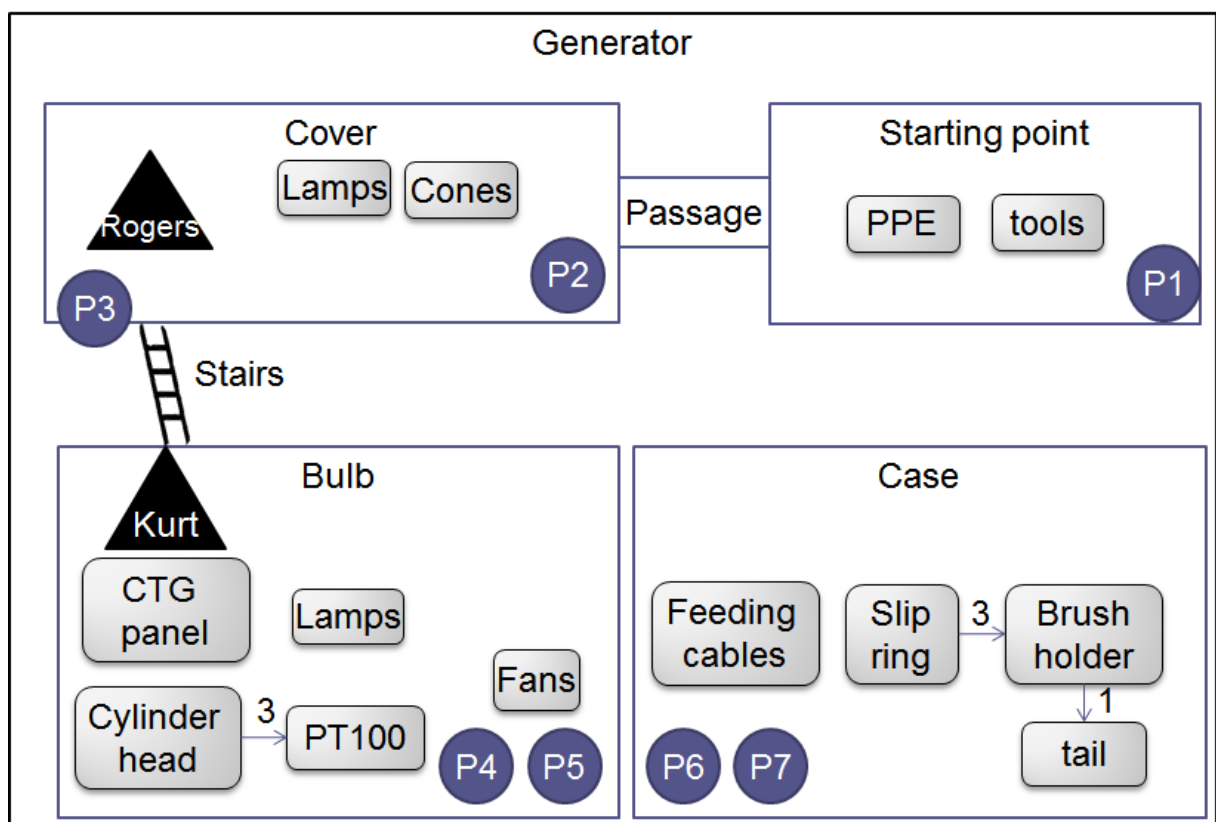


Figure 6.8: Schematic of the entities (agents) present in the study case/simulation. It shows agents of the type Environment (rectangles), Equipment (rounded rectangles), Procedure (circles) and Team members (triangles).

Personal agents were not created because there was not a list of users (technicians) of the application beforehand. System agents were not created because they were not necessary for this task.

6.4.1.4 SA Model

Five Specialized Decision Components were created for this simulation. All are presented in Appendix B.

6.4.1.5 Presentation components

The only presentation component created was already shown in Chapter 5.

6.4.2 UI Design

In the following sections, each decision established in the GDTA of Figure 6.4 will be defined in agents through Decision Components, so that the run-time can generate the SASUI to support users during the simulation.

The initial focus of the design process was on modeling Local Awareness (Procedure Awareness, as discussed in Section 4.4), thus the agent GI. To facilitate this process, as stated in Section 5.5.1, a connection between HTA and GDTA was made, defining if each task (HTA) had a correspondent decision (GDTA). These decisions **will be the first to be modeled**, followed by all the other in their numbered order.

Decision 1.2: How to inspect the equipment?

This decision was modeled using the Procedure inspection SpecDC (Appendix B). Each of the 16 inspection procedures was defined using the four variables of the Procedure inspection SpecDC: sensing possibilities; nature of the problem; quantity of equipment; cue of the problem.

Figure 6.9 shows the choice of variables and the chosen UI for each of the 16 procedures. Each procedure was linked to their corresponding Equipment agent, with the exception of cleanliness and noise inspections, which were linked to an Environment agent. Cleanliness was duplicated and linked to two Environment agents, Cover and Bulb. Noise was linked to the Environment agent Bulb.




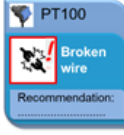
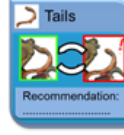
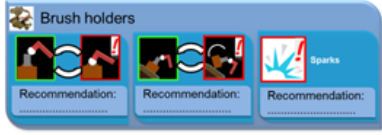

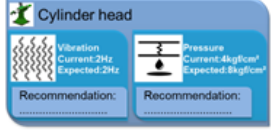




Categories	Procedure number and UI		
I: indirect sensor D: defect G: group of equipments V: visual	2 and 4	4	
			
O: organizational A: abnormality E: environment (no source) V: visual	2 and 4		
			
O: organizational D: defect G: group of equipments V: visual	5	7	
			
O: organizational A: abnormality G: group of equipments V: visual	7		
			
S: sensor A: abnormality E: environment (no source) A: auditory	4		
			
S: sensor A: abnormality U: unique V: visual	5	6	
			
S: sensor A: abnormality G: group of equipments V: visual	5	6	7
			

Figure 6.9: Inspection procedures definition for the study case, for Decision 1.2: How to inspect the equipment?, using the Procedure inspection SpecDC (Section 5.6.1).

Decision 1.3 Is it necessary to fix the equipment (categorize the type of maintenance)?

This decision was defined using the Inspection Maintenance Classification SpecDC. Table 6.3 shows the list of problems and their cost, risk and impact. Most problems have a huge impact, because the whole generator system needs to be stopped for repairing them.

Table 6.3: Cost, risk and impact of every problem of the study case.

Problem	Cost	Risk	Impact
Lantern not working	1	1	None
Broken lamp in the cover	2	1	None
Cleaning problem in the cover	1	1	None
Broken lamp In the bulb	2	1	None
Cleaning problem in the bulb	1	1	None
1 broken fan	10	2	Generator stop
2 broken fans	10	5	Generator stop
Abnormal noise	10	3	Generator stop
Vibration in the cylinder head	10	4	Generator stop
Oil leak in the cylinder head	10	2	Generator stop
PT100 with high temperature	10	4	Generator stop
1 PT100 with broken wire	10	2	Generator stop
All PT100 with broken wire	10	5	Generator stop
Overheat feeding cable	10	2	Generator stop
Overheat of slip ring	10	3	Generator stop
Vibration in the slip ring	10	3	Generator stop
Spark in the brush holder	10	2	Generator stop
Brush wear	10	2	Generator stop
Brush holder misaligned	10	2	Generator stop
Overheat in the tail	10	2	Generator stop
Broken wire (tail)	10	3	Generator stop

Table 6.4 shows associated problems and the cost and risk when they occur together.

Table 6.4: Cost and risk when problems occur in combination.

Combination of problems	Cost	Risk
Oil leak in the cylinder head	10	5
Vibration in the cylinder head		
Vibration in the cylinder head	10	5
Vibration in the slip ring		
PT100 with high temperature	10	5
Overheat of slip ring		
PT100 with high temperature	10	5
Overheat of slip ring		
2 broken fans		
Abnormal noise	10	4
Overheat feeding cable		

Using both tables, the run-time architecture generated the UIs shown in Figure 6.10, Figure 6.11 and Figure 6.12 for respectively situation S1, S2 and S3 of the simulation (Table 6.1).

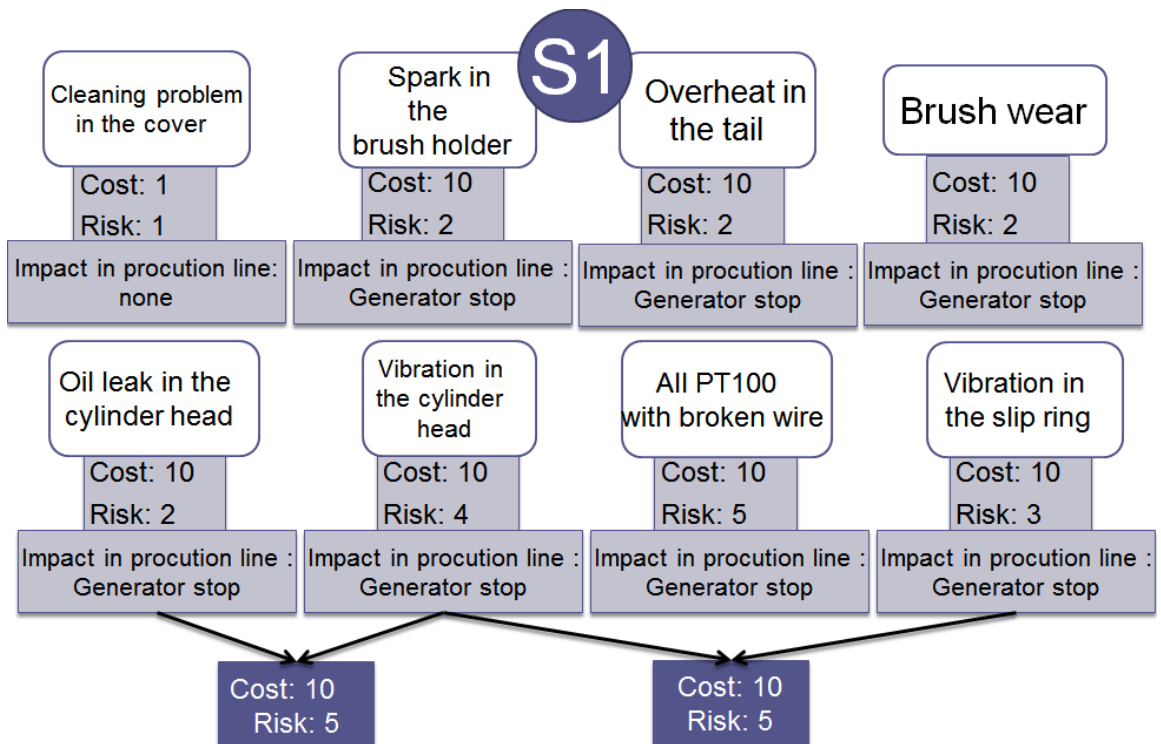


Figure 6.10: UI generated for situation S1 of the simulation for the Decision 1.3 Is it necessary to fix the equipment (categorize the type of maintenance)?.

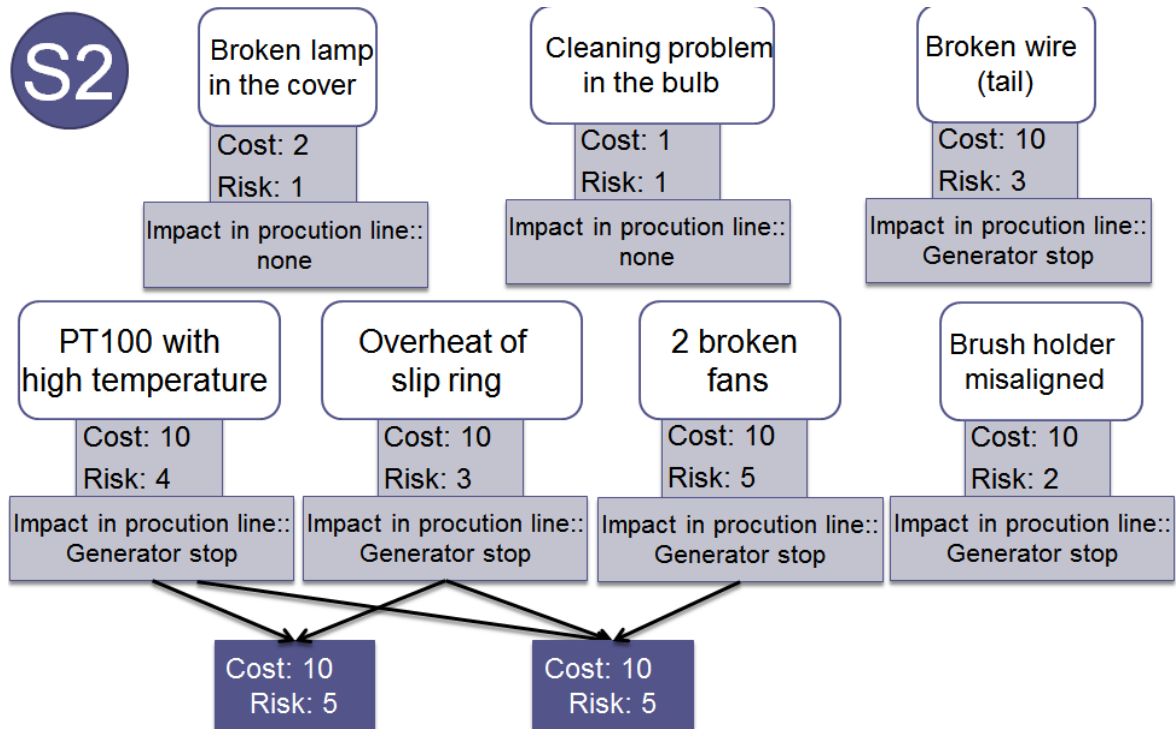


Figure 6.11: UI generated for situation S2 of the simulation for the Decision 1.3 Is it necessary to fix the equipment (categorize the type of maintenance)?.

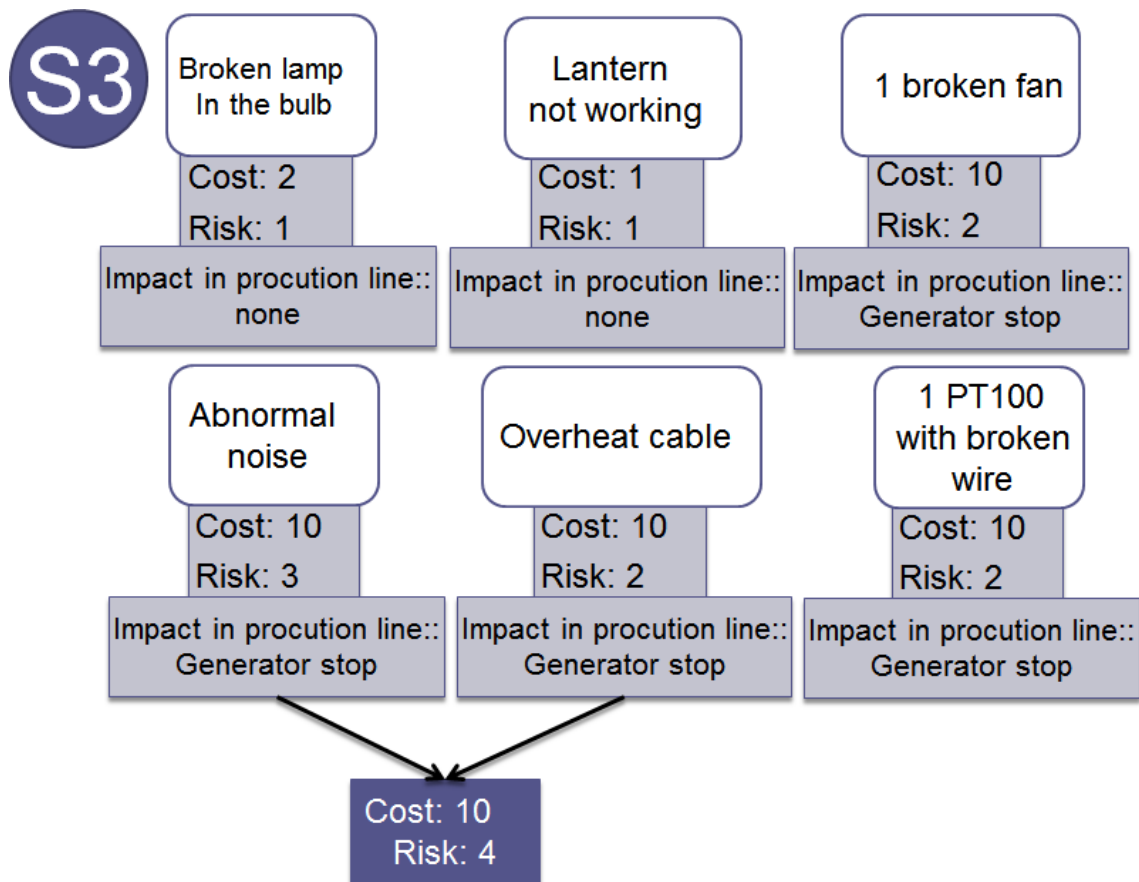


Figure 6.12: UI generated for situation S3 of the simulation for the Decision 1.3 Is it necessary to fix the equipment (categorize the type of maintenance)?.

Decision 2.1 What are the risks to be prevented?**Decision 2.2 What equipment / tools need to be used to prevent personal injury and environmental risks?****Decision 2.3 Is it safe to start operating the equipment?****Decision 2.4 Which tool is required to execute the work?**

These decisions were modeled together using two Personal Protective Equipment SpecDC (Appendix B). The output UIs of the Decision Components are displayed in Figure 6.13. Each Equipment agent necessary was dully linked to this Decision Component.

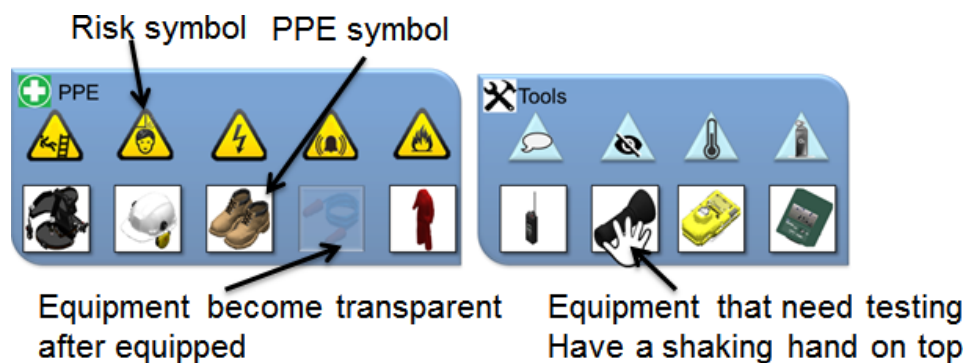


Figure 6.13: (A) UI defined for Decision 2.2 What equipment / tools need to be used to prevent personal injury and environmental risks? and Decision 2.1 What are the risks to be prevented?; (B) UI defined for Decision 2.4 Which tool is required to execute the work?; Both UIs used the Personal Protective Equipment SpecDC (Section 6.4.1.4.1).

Decision 2.5 Where should the Collective Protective Equipment be deployed?

For this decision, the Collective Protective Equipment SpecDC was used (Appendix B). The CPE adopted were cones that needed to be positioned in the generator cover room, next to the stairs. Figure 6.14 shows the component UI.

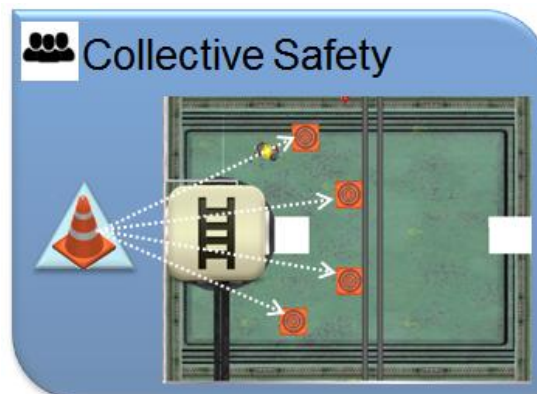


Figure 6.14: UI defined for Decision 2.5 Where should the collective protective equipment be deployed?, using the Collective Protective Equipment SpecDC (Section 6.4.1.4.2).

Decision 1.1 Which equipment should be inspected?

For this decision, four Decision Components were created and reused for every inspection. Table 6.5 shows these Decision Components and the agents that have them.

Table 6.5: Decision Components created to model the Decision 1.1 Which equipment should be inspected?, and the agents that have these components.

Decision Component ->	One sensor	Two sensor	No sensor but regulated	Organization
Agents ->	Lamp (cover)	Cylinder Head	Brush holder	Cleanliness (cover)
	Lamp (bulb)	Slip ring		Cleanliness (bulb)
	Fan			
	Noise			
	PT100			
	Feeding cable			
	Tail			

Exploring the modeling of the “One sensor” Decision Component, after the GDTA adjustments, the resultant GDTA became:

- 3- Impact of not doing the inspection
- 2- Expected condition of the equipment
 - 1- Last inspection performed on the equipment
 - 1- Abnormalities list by the operator
 - 1- Current sensor measurement
 - 1- Expected sensor measurement
- 1- Regulation
 - 1- Law (regulatory standards)
 - 1- Company policies

The high-level trigger defined for these components is the Start Task one, so that this component is activated when a task starts. Each component then used one of the procedures (1 to 8) defined in the Procedure agent as their trigger.

DFAs of levels 2 and 3 of this component were already discussed in Section 5.5.3.3. No level 2 or 3 SAW dynamic equation was defined.

The UI defined is shown in Figure 6.15. This UI focus on describing the current condition of the equipment, based on past reports and sensor measurements, the regulations bounded to the equipment, and the impact this inspection has (importance).

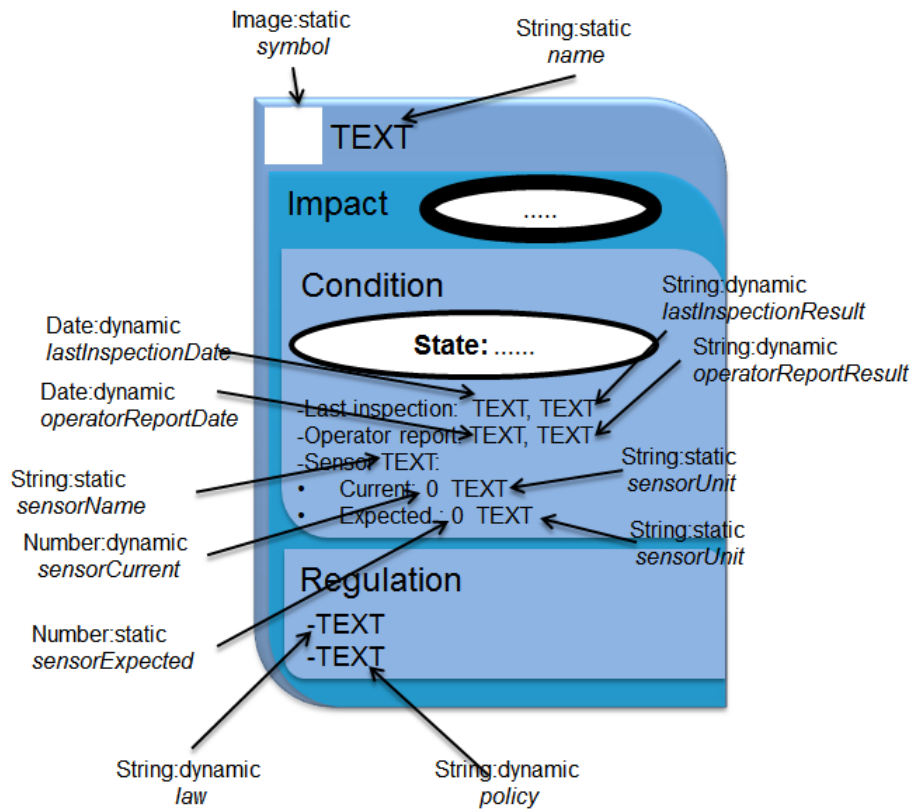


Figure 6.15: UI components with information for the Decision 1.1 Which equipment should be inspected?.

Then, static information was defined, as shown in Figure 6.16 (marked in bold).

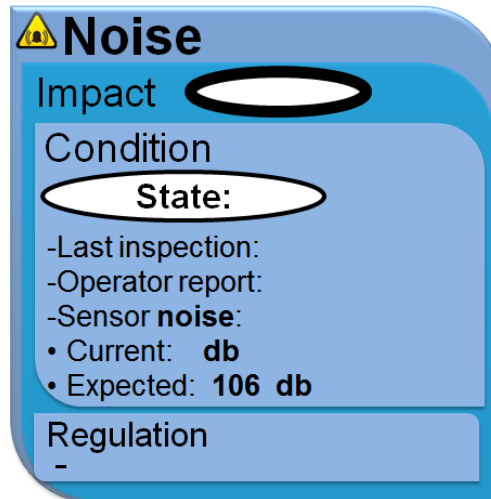


Figure 6.16: Static information defined for the UI component of the Decision 1.1 Which equipment should be inspected?.

UIs for each state of the level 2 and 3 DFA were also defined. Each state changed the impact or condition field in the UI of Figure 6.16, resulting in Figure 6.17.

Concluding, transition Messages were created with the intention of communicating with their Procedure agent counterpart. Each inspected Equipment agent had this Decision Component to inform the user more details about the inspection, and the “How to inspect the

equipment” Decision Component of the agent GI. To assist this other component, this one sends messages to increment the other UI, or switch it to the Nonvisible mode in case the equipment is in a perfect state. This way, users could actually skip a few of the procedures (for an equipment that was interpreted as in a fine functioning state).

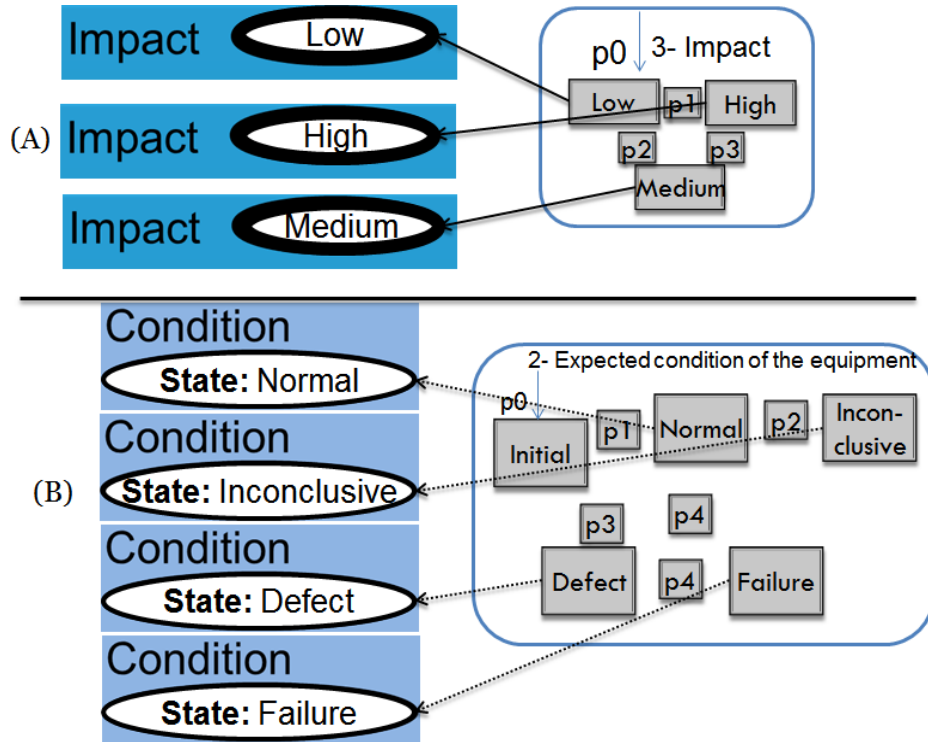


Figure 6.17: UI transformations for each DFA state for the Decision 1.1 Which equipment should be inspected?, with (A) level 3 of SAW and (B) level 2 of SAW.

Table 6.6: Transition Messages schematics of the Decision 1.1 Which equipment should be inspected?.

Transition	Mechanism (action)	Target	Properties
L2-p1	Pass information to agent	Proc.Current.Target	“Recommendation: ignore”
L2-p1	Switch agent to Nonvisible mode	Proc.Current.Target	
L3-p1 e L3-p2	Pass information to agent	Proc.Current.Target	“Recommendation: inspect”

The other components were created as variations from this one, with few differences in the DFA transitions.

Decision 2.6 What are the possible risks to be aware?

In the GDTA, three types of distinct information requirement were listed for this decision. The first was related to risks by time, which are environments that become unhealthy after a certain amount of time (insalubrity).

The second is risk by actions, which happens when a certain action needs to be executed, like using a stairs, or operating dangerous equipment.

The third is risk by presence, which is staying in an environment with a high possibility of risks, like working on heights, or in a confined space.

For this study case, two types of risks were modeled, risks by time and action. The first risk modeled regards the WBGT/noise. Although this is a risk offered by the environment, the entity more concerned with it is the Regulatory Norms, so this risk was modeled in the Enterprise agent Company.

Information requirement for this decision (GDTA adjustments) was:

- 3- Best time to take a rest
 - 2- Time remaining to deadline
 - 1- Time since the beginning of the task
 - 1- Value of the pertinent variable (noise)
 - 1- Maximum time of allowed work
 - 1- Time to leave the place
 - 1- Estimated time of interruptions
 - 1- Task total duration
 - 1- Remaining task duration time
 - 1- Shift total time
 - 1- Remaining shift time
 - 1- Procedure interruptibility

The high-level trigger defined was the Area Enter one, so that this component is activated when users enter in the generator area.

Two DFAs were defined, one for the level 2 “2- Time remaining to deadline” and one for the level 3 “3- Best time to take a rest”. Figure 6.18 shows both DFAs.

In the level 2 DFA, transition to the “Leave Now” state occurs when the time the user is inside the generator is higher than the time he/she is allowed to be there according to RN-15 and WBGT/noise demands.

In the level 3 DFA, state transition is more subtle, defined by the transitions:

- p1: if the time allowed to stay is smaller than the time inside, summed with the time to leave and the remaining duration of procedure or shift, then transition to “Leave when possible” state;
- p2: if the procedure is about to finish, the shift is not over and the procedure is categorized as non-interruptible, than transition to “Do not disturb” state;
- p3: if the WBGT value is too small to be considered a risk, than transition to “Do not disturb” state;
- p4: if there is an interruption, and the time allowed to stay is smaller than the time inside, summed with the time to leave, the remaining duration of procedure or shift and the remaining duration of the interruption, then transition to “Leave when possible” state;

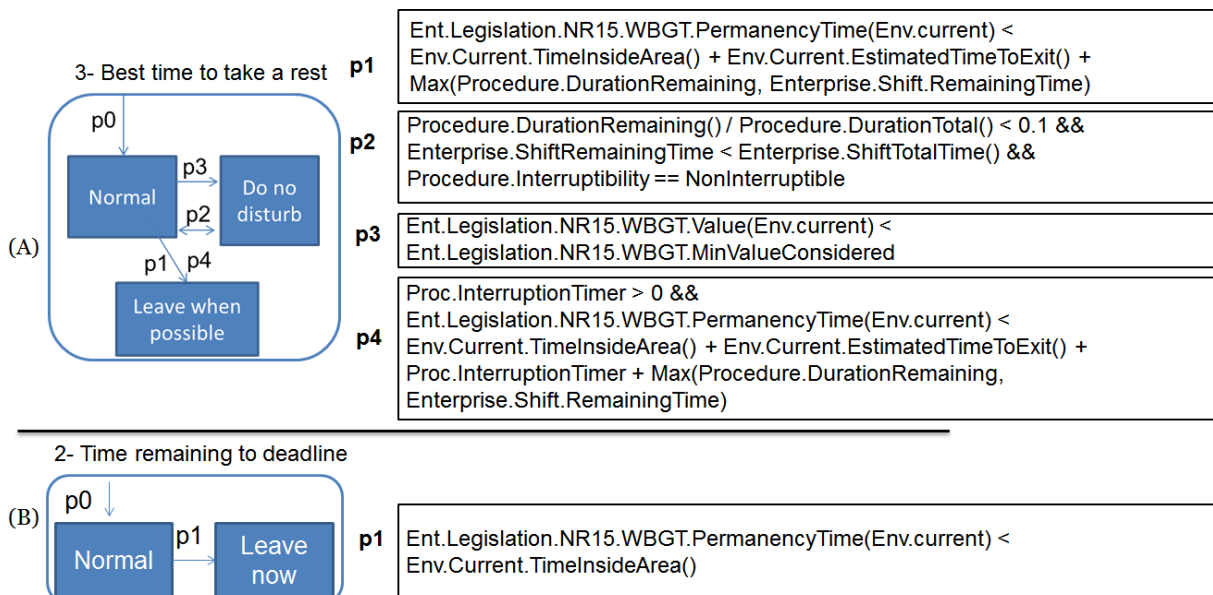


Figure 6.18: DFAs for the Decision 2.6 What are the possible risks to be aware? regarding insalubrity risks.

A level 2 SAW dynamic equation was defined for the level 2 “2- Time remaining to deadline” as:

$$Ent.Legislation.NR15.WBGT.PermanencyTime(Env.current) - Env.Current.TimeInsideArea()$$

The UI of this Decision Component is showed in Figure 6.19, with UI component information in (A) and static information defined in (B).

Finally, UIs for each state and transition Messages are showed in Figure 6.20. The state “Leave when possible” uses a pointing arrow to call user attention, while the state “do not disturb” removes almost every UI component to be more discreet.

Regarding transitions, L2-p1 and L3-p1 both create alarms to warn the user, while L3-p4 goes further, visually connecting information to assist the user in understanding that there is an interruption and it will consume more time than the user has left before the WBGT rest.

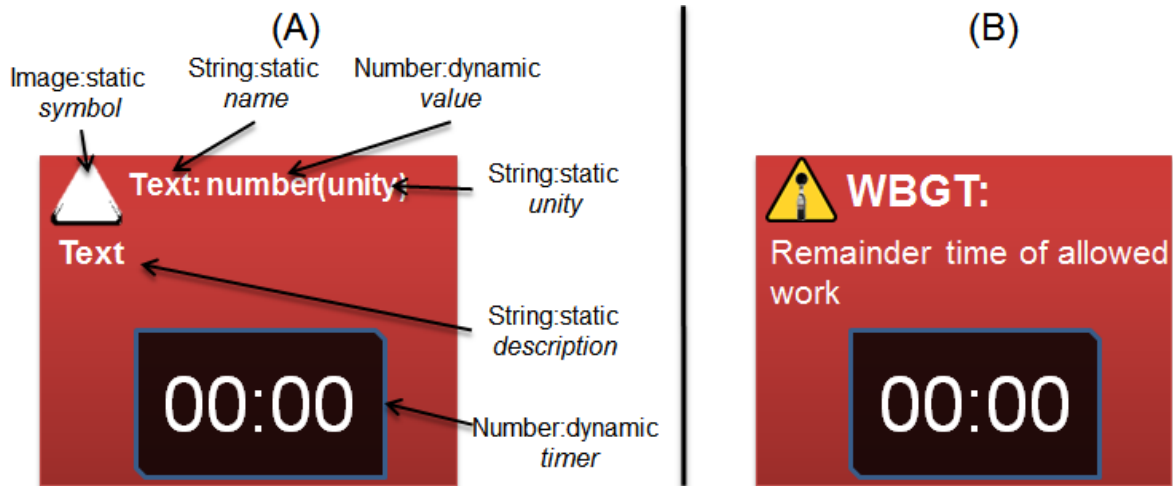


Figure 6.19: UI for the Decision 2.6 What are the possible risks to be aware?. In (A) only UI components and in (B) static information.

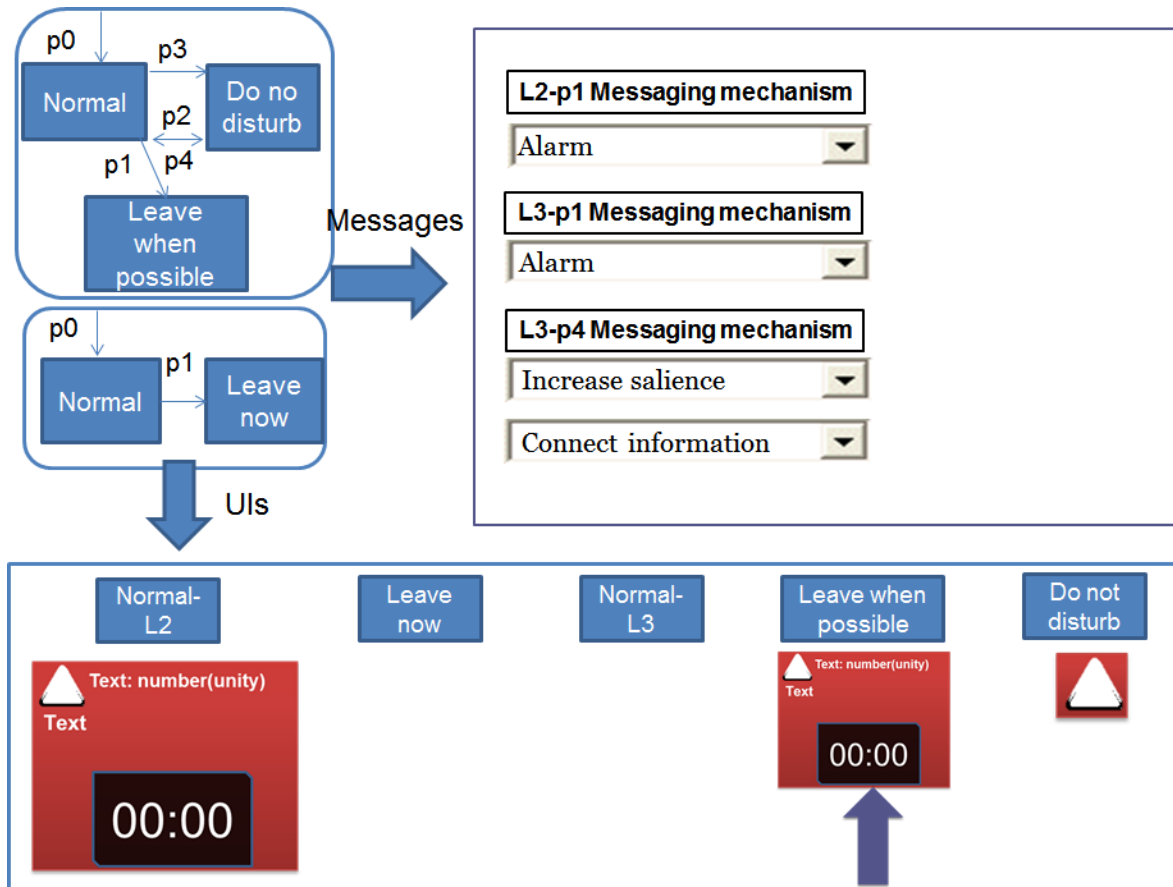


Figure 6.20: Messaging mechanisms and UIs defined for each state of each DFA for the Decision 2.6 What are the possible risks to be aware?, considering the WBGGT/noise as a risk.

Considering a user doing procedure 4, if the L3-p4 transition happened, the whole UI would look similar to Figure 6.21. In this situation, the user needs to work in the cylinder head, but teammate Kurt is working there for 6 more minutes. At the same time, the user needs to rest in 5 minutes. This situation happens in the simulation in situation S3, shown in Table 6.1.

The same Decision Component was then reused (as a PromDC) in the same agent to model noise as a risk.



Figure 6.21: View of the whole UI in Decision 2.6 What are the possible risks to be aware?, when situation “Leave when possible” is triggered by transition p4 (someone is interrupting the user).

The second risk modeled for this decision is related to the stairs (risk by action). This one was modeled in the Environment agent Stairs. The adjusted GDTA for this risk was:

- 2- Protection mode
 - 1- Risk classification
 - 1- Source
 - 1- PPE/CPE

Figure 6.22 shows for this Decision Component the level 2 DFA, the UI for each state and transitions with their mechanisms. Transition p1 happens when the user is close to the stairs, triggering an alarm and a change in procedure, switching temporally this agent to the Procedure space of the UI layout (Figure 4.21). Transition p3 then switches the procedure back to the normal, after the user has distanced from the stairs.

Inspect panel CTG and cylinder head (Kurt)

	P1	P2	P3
Blocking	parallel	parallel	blocking
Risks to others	none	none	none

Figure 6.23: Defining the Decision 3.1. What are my teammates doing that I need to be aware of?, using the Team inspection SpecDC.

Decision 3.2 Is there an incident principle?

For this decision, three Decision Components were created in 3 Environment agents.

The first Decision Component was responsible for fires and was modeled in the Environment agent Generator. It began using the following SAW requirements:

- 3- Projected risk of the environment
 - 2- Current condition
 - 1- Sensor measurement
 - 1- Safety Boundary
 - 1-Value
 - 1-Direction (above/below expected)
 - 1- Trend of the sensor (increase/decrease)

Then, a level 2 and a level 3 DFA were defined, as shown in Figure 6.24. The level 2 DFA had one transition, which happened when the sensor was below or above a safety boundary. The level 3 had a similar transition, but instead of considering the current value of the sensor, it considered a projected value.

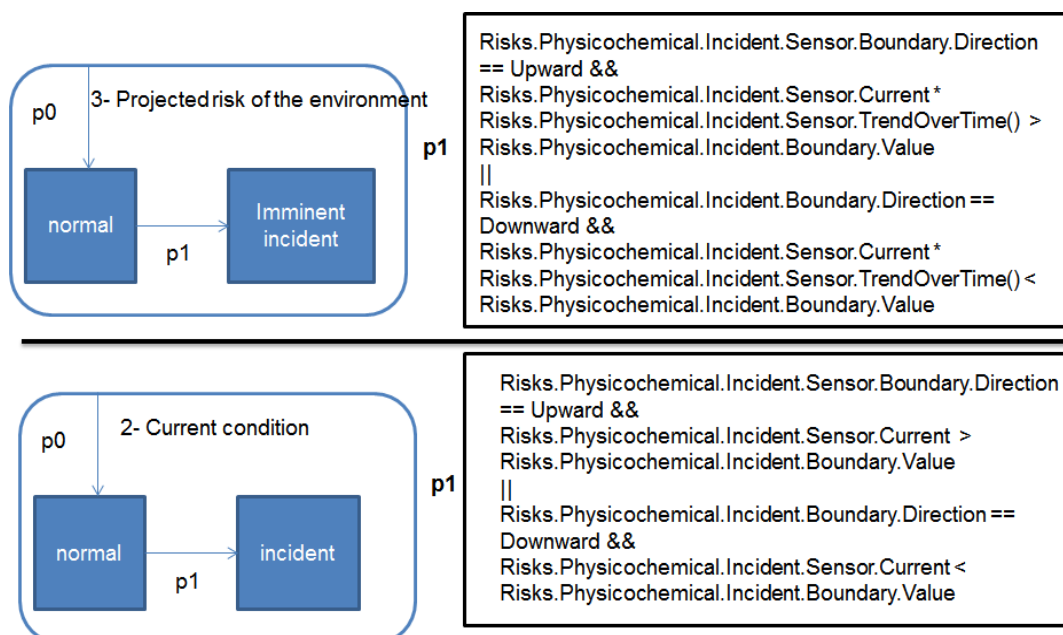


Figure 6.24: DFA for the Decision 3.2 Is there an incident principle?, considering fire.

UIs for this Decision Component and transition Messages are showed in Figure 6.25. As soon as DFA level 3 transitions to state of “imminent incident”, an alarm is fired and the evacuation procedure is triggered. This alarm is reinforced when level 2 DFA reaches the state of “incident”.

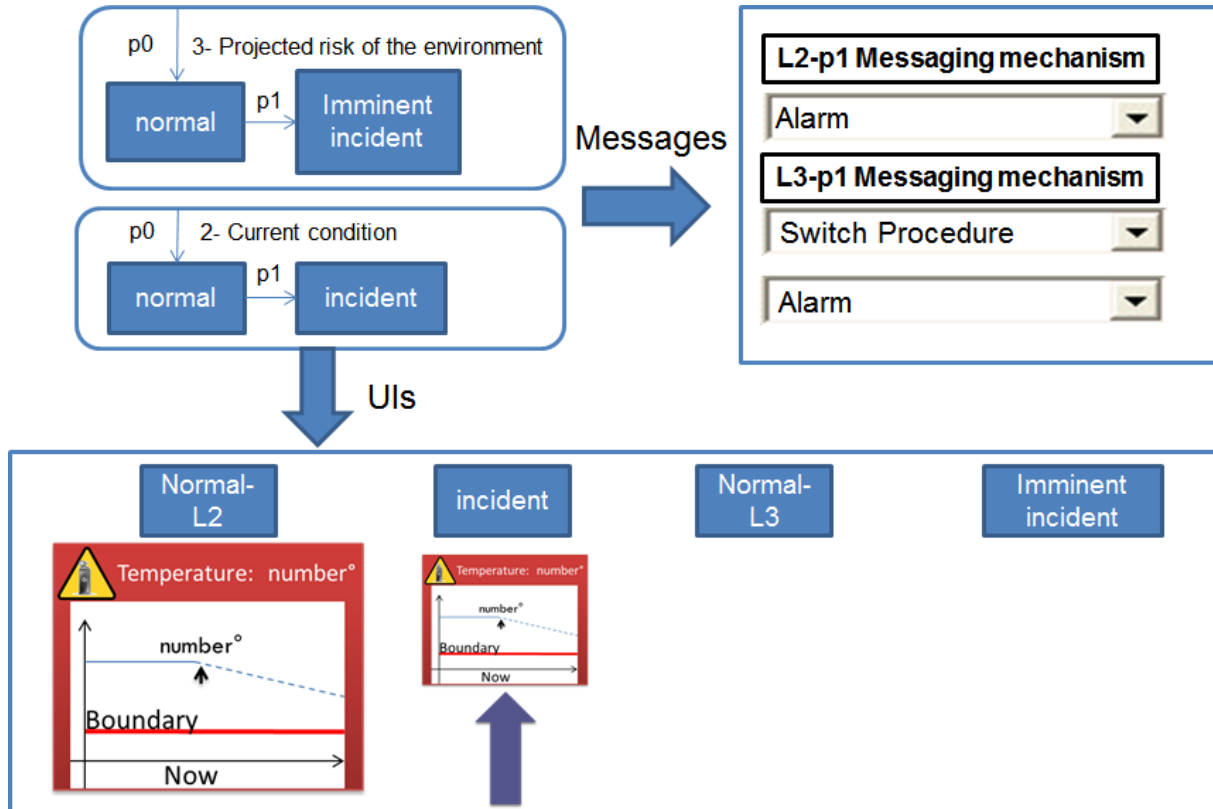


Figure 6.25: Messaging mechanisms and UI for each state of the DFA for the Decision 3.2 Is there an incident principle?.

The same Decision Component was then reused (as a PromDC) in the Environment agent of Bulb and Case, to model the risk of low oxygen.

Decision 3.3 When is the time for a shift handover?

For this study case, this was modeled as a simplistic decision, because many companies do not permit overtime under any circumstance. For others companies, this decision may have extra layers of complexity.

Also, this decision is not impactful in the simulation, because users would always start the simulation in the beginning of the day, the simulation took no more than 30-60 minutes to complete, and their shift lasts for 8 hours. Thus, a shift handover would never happen in the simulation. Nevertheless, it was added to the simulation to increase realism.

The Decision Component was based on the following adjusted GDTA:

2- Best time for shift handover

1- Total shift time

1- Current shift time

The DFA for this Decision Component, along with the UIs of each state and also transitions is showed in Figure 6.26. Transition p1 happens when the user shift is about to end, alarming him/her, and p2 happens when the shift time is over.

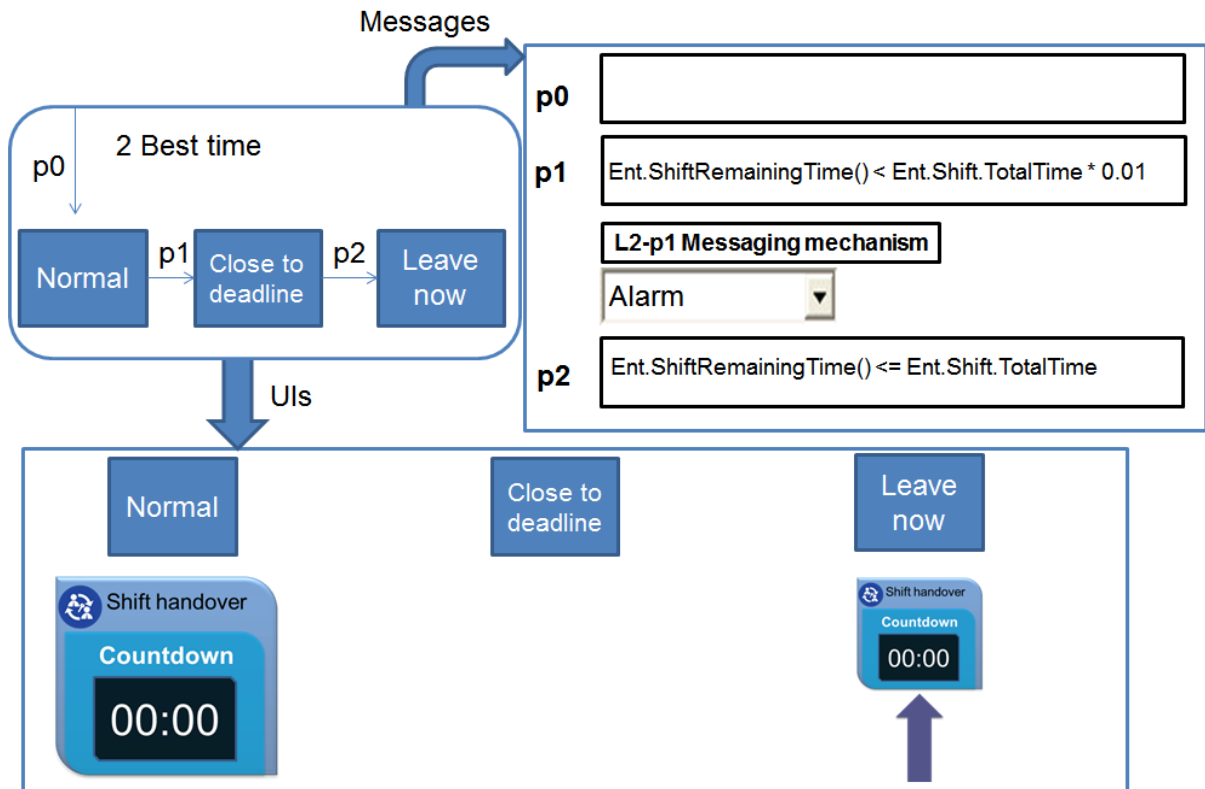


Figure 6.26: DFA, Messaging mechanisms and UIs for the Decision 3.2 Is there an incident principle?.

6.4.3 Iteration after SAW Measurement

SAW measurement and evaluation of SAW-UI was done with experts and is explained later in this chapter. From this evaluation, some improvements were made, generating SAW-UIv2, which was only tested by non-experts.

The most important change was the way it handled the coworker requesting rescue, which happened in situation S3 during the AOC "Gas Leak". To decrease the error rate associated with this AOC, SAW-UI "v2" focused more on Team Awareness, to rationalize that their coworker needed assistance.

Therefore, Team Inspection SpecDC was improved to have a Messaging schematic that alarm users when teammates are feeling sick. This was already described as part of this Specialized Decision Component in Appendix B, but it was actually only done after initial testing, as an iteration of the UI. With each new test, more increments or changes in the UI are

easily done, only adjusting Decision Components of the agents, and when necessary, creating new agents.

6.4.4 Compilation of agents and Decision Components

The final compilation of the UI created for this simulation is summed up in Table 6.7. For this study case, 29 agents were created and 24 Decision Components.

Table 6.7: Compilation of agents and Decision Components of the study case.

Awareness	Agent name	Decision Components
Procedure	Generator	How to inspect the equipment?
	Inspection	What equipment / tools need to be used to prevent personal injury and environmental risks?
		Which tool is required to execute the work?
		Where should the Collective Protective Equipment be deployed?
		Is it necessary to fix the equipment (categorize the type of maintenance)?
Equipment	PPEs (5 agents)	
	Tools (4 agents)	
	Lamp (cover and bulb)	Which equipment should be inspected?
	Fan	Which equipment should be inspected?
	Cylinder Head	Which equipment should be inspected?
	PT100	Which equipment should be inspected?
	Feeding cables A/B	Which equipment should be inspected?
	Slip Ring	Which equipment should be inspected?
	Brush holder	Which equipment should be inspected?
	Tail	Which equipment should be inspected?

Awareness	Agent name	Decision Components
Environment	Generator	Is there an incident principle? (fire)
	Starting point	
	Passage	
	Cover	Which equipment should be inspected?
	Stairs	What are the possible risks to be aware?
	Bulb	Which equipment should be inspected? (noise)
		Which equipment should be inspected? (cleanliness)
		Is there an incident principle? (oxygen)
	Case	Is there an incident principle? (oxygen)
Team	Eagle	What are my teammates doing that I need to be aware of?
Enterprise	Company	When is the time for a shift handover?
	Regulatory norms	What are the possible risks to be aware? (WBGT)
		What are the possible risks to be aware? (noise)

6.5 Study case evaluation

This section explains the experiment to evaluate both UIs regarding their support for SAW. Experts (current technicians from a powerplant) and non-experts were invited to execute the simulation with SAW-UI and Guide-UI. Participants' efficacy was calculated computing their successes and errors in decisions. After the simulation, participants answered a questionnaire, which was evaluating their SAW subjectively (Situation Awareness Rating Technique or SART) and aspects of the usability of the UI.

6.5.1 Participants

Eighteen people participated in the evaluation, 16 men and 2 women. Participants were of two kinds: experts and non-experts (novices).

Ten participants were experts, with at least 5 years of experience (average of 10 years), having executed this procedure in real life at least 5 times, and having full knowledge of all the Regulatory Norms. However experts had median experience with computers and low with games (and some had trouble controlling their avatar).

Of the eight non-experts participants, all of them were very experienced with computers, and 6 were experienced with games (with only two being novices in games). Also all of them had never had any contact with maintenance and hydropower plant inspections before.

It was only possible for non-experts to use the simulation because the difficulty of the simulated work was streamlined to simple “point and click” actions. This means that procedures that usually requires physical training, like wearing latch belts to climb stairs, could be done with a click, and procedures that require mental training, like understanding an equipment condition, could be execute through pictorial comparison of normal versus broken equipment (equipment problems in the simulation were also visually exaggerated to facilitate).

None of the participants had prior experience with the experimentation setup used in this experiment.

6.5.2 Apparatus

Simulations were executed through a desktop computer with mouse and keyboard as input devices. The avatar in the simulation was controlled like a first-person game. Users that had no experience with this control mode were given quick lessons before starting the experiment, and also explicitly told that time was not a factor (thus they did not need to get nervous over their difficulty controlling the avatar).

The inspection task was made through a UI that could be invoked at any time by pressing the space button. Users would inspect the equipment and report abnormalities/defects through this UI, shown in Figure 6.27. After the procedure is finished, users had to manually select to advance to the next activity.

The few interactions the simulation required beyond inspecting and reporting were implemented through a target and click strategy. For instance, when users needed to get Personal Protective Equipment (PPEs), they pointed directly at one and clicked with the left mouse button to get it. To attach a belt, they would point at the chain and click with the left

mouse button. Whenever they pointed at an interactable object, the central aim would turn green (from white).



Figure 6.27: UI to report inspected problems in the simulation.

Therefore, even without any assisting UIs (Guide-UI or SAW-UI), users could complete the task (if they had it memorized). This means that any UI would only assist, but not be mandatory for the simulation.

Finally, expert users were trained in both UIs presented in Chapter 6, Guide-UI and SAW-UI. Non-experts were only trained and used SAW-UI.

6.5.3 Variables of the study and Experimental Design

Regarding the evaluation, one dependent and one independent variable were considered in the study. Three situations were designed for the simulation, each with different AOCs happening (thus different events and with entities having different conditions). Therefore the first variable, dependent, was that for each designed simulation, experts would use Guide-UI or SAW-UI, and the difference between results was considered, to analyze if they could outperform themselves using SAW-UI.

This type of experiment is defined as a randomized complete block design (RCBD) and situations distribution had to be randomized to avoid a leaning effect. Thus an order for executing each situation and a supporting UI was assembled, as shown in Table 6.8.

The first row indicates the execution order. The second row is one of the three situation designed (Table 6.1). The last row is the UI used in the simulation. With this amount of randomization, users could not memorize situations and so each execution felt like the Abnormalities in Operative Conditions were randomly drawn.

Table 6.8: Randomization of situations in the simulation and the UI for each user playthrough to create a randomized complete block design.

Execution order	1	2	3	4	5	6	7	8
Situation	Learning	Learning	1	2	3	2	1	3
UI	Guide	SAW	Guide	SAW	Guide	Guide	SAW	SAW

A **paired t-test** was then applied to compare both UIs and analyze statistically significant differences in the defined metrics (SAW and efficacy).

The second variable, independent, was a comparison between experts and non-experts results metrics (SAW and efficacy). Non-experts would be trained in SAW-UI and the task in S1, and then execute S2-3. An **independent t-test** was executed to analyze statistical difference between both groups of users.

6.5.4 Experimental Procedure

Evaluation was made first with experts and then with non-experts. Experts' evaluation consisted of a comparison of the two UIs (Guide-UI and SAW-UI). The goal of their evaluation was comparing UIs (SAW and efficacy), validating the fidelity of the simulation and receiving feedback on the validity of SAW-UI for their real work.

Posteriorly an evaluation with non-experts was made, but this time only using SAW-UI. The goal was not to compare both UIs, but to introduce the activity and measure performance when users do not completely understand the problems presented (have no mental model of the situations).

Three forms of evaluation were designed: automatic logging of the **efficacy** and **efficiency** for each procedure (indirect evaluation based on performance) and a SART questionnaire (subjective **SAW** evaluation).

We programmed 24 Abnormalities in Operative Conditions (AOCs) in the simulation using a testable response strategy [Endsley12, Pritchett00]. This means these problems would present themselves to users and their response would be verified against the established right response, to compute efficacy.

Error detection was automated in the simulation, allowing for a faster and more accurate testing, compared to direct observation or recording analysis. Nevertheless, although it would be interesting, errors were not classified according to SAW level (1- perception, 2-comprehension and 3-projection). It is hard to analyze committed errors corresponding to a

faulty SAW level in an automated way (the non-automated solution would be to explicitly ask users to justify their actions, using a Think-aloud protocol).

For instance, take an error of not reporting (or incorrectly reporting) an overheating equipment (over 70°) and analyze examples of the 3 levels of SAW that could result in it:

- Perception (level 1): users could simply mistake the temperature value (it was 50° Celsius, they thought they saw a different number on the sensor).
- Comprehension (level 2): users could confuse the value and its meaning, such as confusing Celsius with Fahrenheit (leading to an incorrect interpretation of hot and cold), or simply confusing the situation (interpreting the value 51°C as hot, when actually only values higher than 70°C would be considered so, or even relate incorrectly this value with another problem, like assuming that since air-conditioning is ok, then equipment temperature must be ok).
- Projection (level 3): users could incorrectly judge the impact/importance of performing or not the inspection and skip the procedure (projecting the procedure as unnecessary because the equipment condition was apparently ok, while a company policy demand an annual close inspections, and the last one was more than one year ago).

Following the simulations, a questionnaire was applied. Experts responded to questions more focused on evaluating their experience and the fidelity of the simulation, and comparing both UIs. Non-experts responded only about SAW-UI, with a focus on usability and evaluating their awareness. General questions were made using a Likert scale ranging from: 2) totally agree; 1) agree; 0) neither agree or disagree; -1) disagree; -2) totally disagree. This questionnaire is presented in **Appendix D**.

Both questionnaires also had a SART (Situation Awareness Rating Technique) on it [Taylor90], a self-rating technique where users respond how comfortable and aware they were. SART evaluation consisted of 14 questions (14D-SART) with users giving a score from 0 to 10 to the demand, supply of attention and ease of understanding of the situations and the UI support. These three elements are the components of SART (demand and supply of attention and ease of understanding) and the topics asked in each one are:

- Demand: instability, complexity, variability and demand of attention;
- Supply: awakesness, concentration, division of attention, spare mental capacity, supply of resources to facilitate attention;
- Understanding: quantity and quality of information, familiarity with the situation, understanding of the situation.

A final question is asked for the user to rate his/her SAW.

The overall SART score is then calculated using the general formula: $SART\text{-score} = U - (D - S)$, where: U = average understanding; D = average demand; S = average supply. The complete formula for SART score, considering questions as Q1 to Q14, is on Appendix D.

6.5.5 Results

Two performance metrics were measured in the experiment. The first is the measure of efficiency calculating mean time of each activity comparing Guide-UI and SAW-UI using a paired t-test. The null hypothesis (equality) was confirmed, with $t\text{-Stat} < t\text{ critical}$ ($0,75 > 1,65$; $P = 0,22$). This means there was no difference in efficiency by using both UIs.

The second performance measure, number of errors, was the best metric to compare both UIs, because of its independence from subjectivity and in users's ability in controlling the virtual avatar. Appendix C shows the complete list of errors, their corresponding lack of awareness, and how many times they were committed by users with each UI

Figure 6.28 shows a compiled number of errors per simulation for experts using Guide-UI and SAW-UI. Overall results were calculated by dividing the number of errors of a person by the number of simulations executed. For Guide-UI the average was 2,7 (SD=0.74) and SAW-UI average was 0,7 (SD=0.57), meaning that users would commit 3,85x less errors using SAW-UI. A paired t-test rejected the null hypothesis, with $t\text{ Stat} < t\text{ critical}$ ($-6,73 < -1,83$; $P < 0,001$), meaning a proven statistical significance in the results.

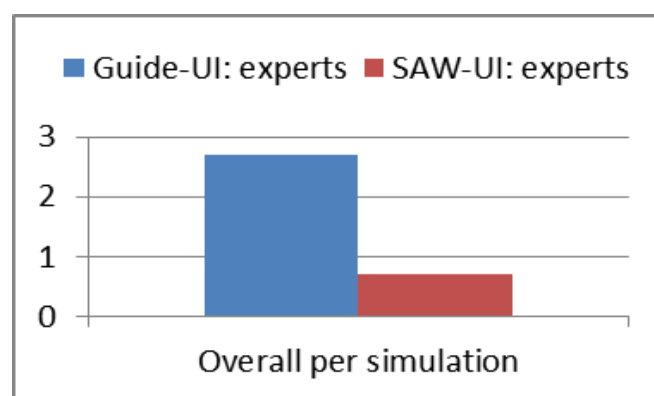


Figure 6.28: Comparison of number of errors committed per simulation using Guide-UI and SAW-UI

Another important metric is safety. To compare safety results, a list of 11 errors that could impact in safeness was compiled (therefore, no inspecting errors). These errors are marked in bold/underline in Table C-1. Comparing results only of these errors, Guide-UI led

users to commit a total of 62 errors (during the 3 simulations) and SAW-UI a total of 16, which is 3,87x less errors regarding unsafe behavior.

Errors can also be classified as Local and Global SAW errors. Figure 6.29 presents information of errors in Local and Global SAW and overall errors, committed per users (expert and non-expert) per simulation using both UIs. As explained in Section 4.2, Local Awareness is Procedure Awareness and Global Awareness is Equipment, Team, Enterprise and Environment Awareness. Each error was classified as a lack of awareness in one or more of these situation aspects of the FSA, as shown in Table C-1. The graphic in Figure 6.29 also express each individual awareness result for Global SAW (Equipment, Team, Enterprise and Environment).

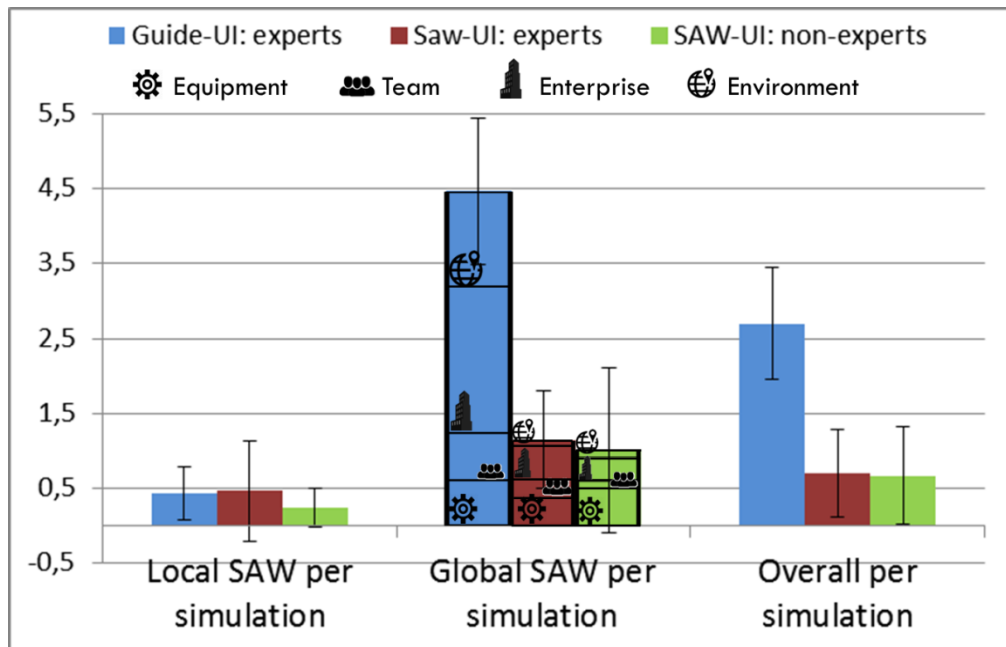


Figure 6.29: Number of errors committed per simulation by users with both UIs, categorized by Local SAW, Global SAW (Equipment, Team, Enterprise and Environment) and overall.

A paired t-test between Guide-UI and SAW-UI was applied. The null hypothesis was confirmed for Local Awareness, with t-test demonstrating equal averages with $t_{stat} > t_{critical}$ ($-0,13 > -1,78$; $P = 0,03$). The null hypothesis was rejected for Global Awareness, with t-test demonstrating different averages with $t_{Stat} > t_{critical}$ ($8,21 > 1,75$; $P < 0,001$). One outlier was removed from the data set, because it was outside the reliability interval ($\mu + 2\sigma$).

Another result from the experiments was the questionnaire (Appendix D). Raw results of SART are presented in Table 6.9.

Table 6.9: SART results compiling the average of answers from experts and non-experts users.

		Guide-UI experts	SAW-UI experts	SAW-UI non- experts
Demand (instability, complexity, variability and demand of attention)	Avg	5,30	5,36	4,39
	SD	2,16	2,53	1,19
Supply (awakeness, concentration, division of attention, spare mental capacity, supple of resources to facilitate attention)	Avg	5,42	7,12	6,31
	SD	2,59	2,07	1,70
Understanding (quantity and quality of information, familiarity with the situation, understanding of the situation)	Avg	5,68	8,51	8,29
	SD	1,82	1,14	1,98
SAW-question	Avg	6,40	8,60	7,5
	SD	1,14	1,67	2,51
SART-final	Avg	5,83	10,42	10,39
	SD	4,41	3,07	3,74

Since Supply is diametrically opposed from Demand, they provide an interesting comparison. Supply (7,12 average; SD=2,06) was higher than Demand (5,36 average, SD=2,53) in SAW-UI, with a paired t-test confirming the statistical difference ($t_{stat} > t_{critical}$; $1,87 > 1,83$; $P = 0,04$).

Another valid comparison is the Understanding component from SART, when comparing both UIs. SAW-UI scored an average of 8,5 and Guide-UI of 5,6. A paired t-test rejected the null hypothesis, with $t_{Stat} > t_{critical}$ ($5,27 > 1,83$; $p < 0,001$), meaning a proven statistical significance in the results.

The final comparison is between SART-final (the final score) of both UIs. A paired t-test was applied between Guide (5,83 average; SD=4,41) and SAW-UI (10,42 average; SD=3,07) rejecting the null hypothesis, with $t_{Stat} > t_{critical}$ ($4,24 > 1,83$; $p = 0,001$), meaning a proven statistical significance in the results.

Finally, a comparison of all the scores of SART (SART-final) is shown in Figure 6.30. Due to the nature of the SART equation ($SART\text{-score} = U - (D - S)$), higher Understanding and Supply results in a better SART-score. An interesting fact from Figure 6.30 is that Non-experts gave the same SART-score as experts for SAW-UI, which is proof of its consistency.

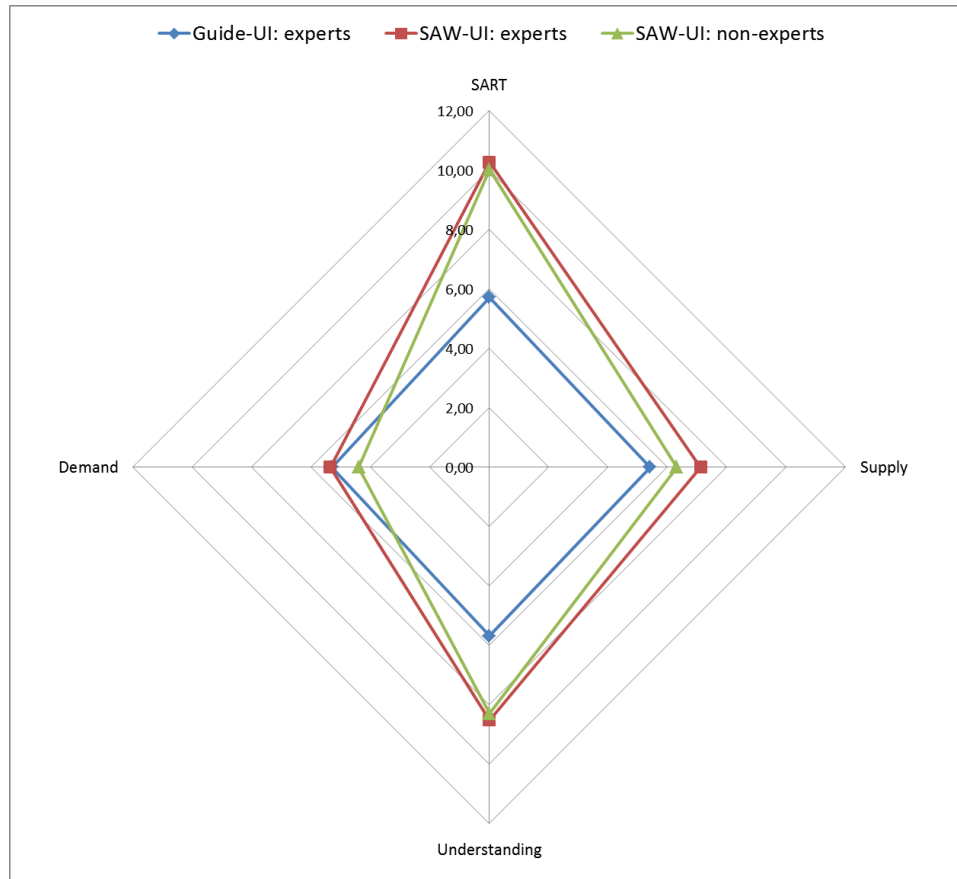


Figure 6.30: SART result for Guide-UI and SAW-UI (experts) and SAW-UI (non-experts).

A final element of the questionnaire was five questions experts were asked related to usability and usefulness of SAW-UI. The questions were answered using the Likert scale:

1. SAW-UI is easy to understand and use.
2. In SAW-UI, it is easy to remember where each type of information is located.
3. SAW-UI would make my life easier than Guide-UI (if properly adjusted to my routine tasks).
4. SAW-UI helps eliminate hazards better than Guide-UI.
5. SAW-UI showed an ideal amount of information (not more and not less).

Every question had a positive result, with the average of 1.34, demonstrating an overall subjective acceptance, as demonstrated in Figure 6.31. Overall usability of SAW-UI was well evaluated combining experts and non-experts, with the majority of answers being either agree or totally agree (average 1.44, $SD=0.76$).

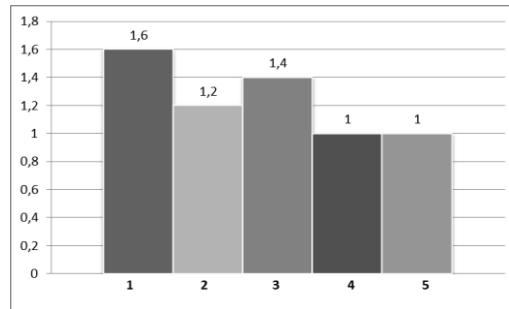


Figure 6.31: Answer to five questions experts were asked related to usability and usefulness of SAW-UI, using the Likert scale.

6.6 Final considerations

This chapter presented the study case used to prove and test the MBSAW-UI methodology with its architecture. First the study case was analyzed thoroughly, with a task, domain and cognitive analysis.

From these analyzes, a simulation was planned and implemented. Three situations were planned for the simulation, to maximize the data collected, such as users' decisions under several different circumstances.

A list of possible errors was defined, and errors were even classified according to their situation aspect (using the aspects defined in the FSA). This classification will facilitate analyzing the differences between Local and Global SAW performance of users.

Then, two UIs were implemented to support users during the simulation: Guide-UI and SAW-UI. Guide-UI was implemented from a Hierarchical Task Analysis of the study case. Then SAW-UI was implemented using the MBSAW-UI methodology, to become a Situation Awareness Support UI (SASUI).

After the implementations, experts and non-experts were invited to test the simulation with both UIs, in a randomized complete block design type of experiment, and results were collected and analyzed.

The next Chapter will discuss the requirements imposed on this project, discussing each of the propositions from introduction (Chapter 1).

Chapter 7

CONCLUSIONS

7.1 Initial Considerations

Situation Awareness (SAW) is an important cognitive process in domains that demand decision-making under dynamic and complex situations and which users are assisted by technological apparatus, such as maintenance and industrial work. As shown in this work, studies in SAW could boost safety and efficacy in maintenance, allowing a better prevention of accidents and an increase in process reliability.

Nevertheless, there were only timid approaches to analyzing SAW in maintenance and industry. Furthermore, this domain consistently ignores the potential of implementing or improving User Interfaces to support SAW.

This thesis focuses on filling this gap between industry needs and Cognitive Engineering recent developments. In order to do so, it deals with SAW and User Interfaces in maintenance work, proposing a methodology to guide the implementation of a Situation Awareness Support User Interface (SASUI). By doing so, this work offers support for developers to develop the UI, and for designers and SMEs to design UIs.

The proposed model based methodology MBSAW-UI facilitates UI design through a high-level process and the MAS architecture accelerates UI development through the autonomous SAW-driven interface agents.

This solution will not only facilitate development, but also allow for a continuous process of User Interface design even long after the system is ready. Therefore, the present End-User Development approach increases applicability of this study, because it is flexible

enough to respond to the rate of changes and the need for continuous improvement currently seen in most industries.

To close this work, the next section is a discussion and analysis of this thesis propositions.

7.2 Discussion of the thesis propositions

7.2.1 P1 - design process

Proposition: the solution proposed concretize a valid vision for design and development of Situation Awareness Support UIs for maintenance work, by attending the quality indicators for SAW support, and the final product (UI) could be directly translated to a real case scenario to improve efficacy and safety of the worker (specially compared to the current systems based on manuals).

This proposition can be divided in two to facilitate discussions: attendance to the quality indicators for SAW support; improving efficacy and safety of the worker.

Regarding the quality indicators for SAW support:

- Cognitive Task Analysis: the MBSAW-UI methodology proposed two forms of CTA that are complementary, the GDTA and the FSA;
- SAW evaluation: the study case was evaluated by the SART technique, and results will be presented throughout this chapter;
- Other evaluation: a performance-based evaluation was also applied, to evaluate efficacy and efficiency;
- Personalization: personalization is possible through Personal agents. For each user of the system, a Personal Awareness agent can be created, with information (in Data Components) about the user. This information could then be used in many Decision Components from many agents to personalize UIs based on user characteristics.

The other indicators are related to the SAOD guidelines and will be explored in P2.C. A discussion regarding the autonomous SAW-driven agents and the state of the art in SAW was already showed in Section 4.5.3.

The second discussion is about improving efficacy and safety of the worker. These factors can be evaluated based on the results of the experiment, and the conclusion from the results is that efficiency was not improved, but efficacy and safety were greatly improved.

As discussed in Section 6.5.5, efficacy and safety can be improved through the use of SAW-UI and its design guidelines, but efficiency was not improved, which was expected because there was little focus on SAW-UI to improve Local Awareness (Procedure Awareness), which would contribute to efficiency.

Efficiency was not improved due to two main reasons may have led to this result: 1) Each person had their own pacing of doing their work (and controlling their avatar); 2) Many of the sensors and information in SAW-UI to assist the inspection procedure do not exist in real life, and so experts trusted their expertise and relied on their senses (especially vision) to decide whether or not the equipment was broken. This resulted in spending the same time to execute inspections, independent of UI.

Finally, in the questionnaire applied for experts, the question “Would SAW-UI make my work easier than Guide-UI?” received an average score of 1,4, ranging from “totally agree” (2) to “agree” (1). Another question, “Would SAW-UI assist better in safety than Guide-UI?”, received an even better average score of 1,5.

7.2.2 P2 - Situation Awareness support

Proposition: the model-based process generates UIs that support users in acquiring and maintaining Situation Awareness.

There was already a brief discussion, in Section 4.4.3, regarding the conformity of the autonomous SAW-driven agents to the state of the art in SAW. This section will complement it, discussing this proposition base on four other items:

- Global SAW: the discussion started in Section 4.2 culminated in a series of strategies created to support designers in developing for Global SAW in MBSAW-UI, and they were evaluated by the study case;
- SAW demons: SAW demons were discussed as an important aspect of Situation Awareness in Section 2.3.1, and they are important topics to be approached when discussing the value of a methodology for SASUI design;
- SAOD guidelines: these were the first and still most important guidelines for SASUI design, and they were absorbed in MBSAW-UI as an integral part of it;

- SA model: an SA model can be a lever for all the other items in this list, facilitating the achievement of a Situation Awareness Support UI or system.

7.2.3 P2.A - Global SAW

Proposition: the final product (UI) assist users in acquiring and maintaining Global SAW while not impairing Local SAW.

Global SAW considering maintenance work was already defined as the Awareness of Equipment, Team, System, Enterprise and Environment. Supporting these awarenesses is essential to improving users fast goal switching and holistic view of the situation.

Many maintenance system UIs, especially because of screen/space limitations, focus only on supporting Local Awareness, showing only procedure information. These can lead users to narrow focus to only procedure, which is the cause of many reported accidents [Sneddon13, Golightly13, Bullemer13].

This thesis solution to supporting Global Awareness is to structure Situation Awareness considering a holistic view of workers situations. The tool created to facilitate this is the Framework of Situation Awareness Aspects (FSA).

GDTA is still the main tool for analyzing cognitive demands of an activity, but since it has absolutely zero point of start, two different designers main generate completely different GDTAs for the same tasks. The FSA is that complementary tool that will assist the formation of a more standardized GDTA and also amplify it.

As a conceptual framework, it give designers the lacking structure (of the GDTA), so that they need to think not only on goals and decisions, but how these correlate with real world entities, and which of these entities are more/less important in the decision-making and work process.

For instance, with a GDTA, designers may conclude that safety is important and describe some decisions related to this topic. With the FSA, entities related to safety are captured and described, and this process will feed into the GDTA, improving the information requirements regarding decisions related to safety.

Both GDTA and FSA create a thinking cycle, in which designers may think of an entity and what decisions are related to it, and think of decisions and what entities are related to it. This cycle is, what we authors believe, a missing link in SAOD [Endsley12].

Regarding the study case and SAW-UI, Chapter 6 shows how a decision-driven design led to considering decisions that impact Local and Global SAW. Chapter 4 discussed the partition dynamism caused by the FSA structure and how it facilitates attention cycling.

These strategies (coupled with the MAS architecture) led to the results that show that Global Awareness can be increased without compromising Local Awareness, as shown in Figure 6.29.

Users were also asked to subjectively rate if SAW-UI helped them understand the 5 situation aspects from FSA (Procedure, Equipment, Team, Enterprise and Environment), and results varied between “totally agree” (2) to “agree” (1) and are presented in Figure 7.1. These results indicate an overall agreement that SAW-UI assisted in all aspects of the situation, with Team Awareness being the worst result (although still positive).

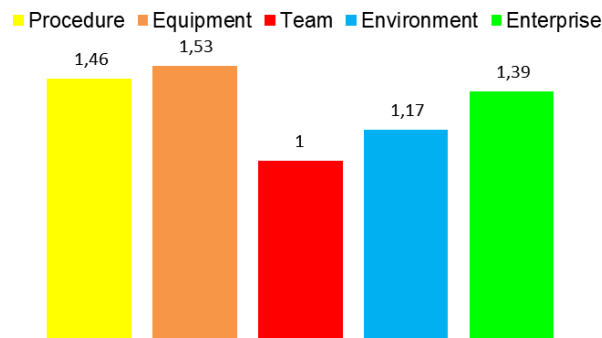


Figure 7.1: Users subjectively rated (Likert scale ranging from 2 to -2) if SAW-UI helped them understand the 5 situation aspects from FSA (Procedure, Equipment, Team, Enterprise and Environment).

7.2.4 P2.B - SAW demons

Proposition: the final product (UI) assists users in avoiding frequent problems with the SAW demons.

Overall, the conclusion is that SAW-UI did manage to improve upon the SAW demons (Section 2.3.1). Each of the 8 SAW demons is analyzed ahead, correlating to the results.

7.2.4.1 Attentional narrowing

Many strategies collaborated for mitigating this factor. A common cause of attention tunneling is associated with data overload, when users cannot absorb more information and start to focus on only one aspect of the system/task. The filtering strategy of the architecture,

with the two layers of information (Visible/Nonvisible modes) contributed to having less information (along with the decision-driven design).

Another point is the partition dynamism and “Big picture” display, which allowed for easy switching of focus during tasks and attention cycling, keeping the user Global SAW up-to-date (evaluated through efficacy).

The final strategy was the cooperation strategy used by the agents (Messaging mechanism), modeling alarms based on the projection of situations, and also modeling mechanisms to trigger procedures for abnormal/urgent situations, such as when the procedure to go down stairs is activated when users are close to it (Section 6.4.2) or when a procedure for emergency evacuation is activated when a fire is started (Section 6.4.2).

Regarding the simulation, two results show the performance to downgrade the attention narrowing problem: the result of a good Global SAW, demonstrating attention division; the question in SART about division of attention had a better score for SAW-UI (8,2 average against 5 for Guide-UI, with a $P < 0,01$).

7.2.4.2 Requisite memory trap

Having the autonomous SAW-driven interface agents assist the users during their tasks alleviated short-term memory. They supplied the necessary information at each moment for the user, which was modeled by the decision-driven design and the adaptations in the UI based on the current situation.

Alarms also worked on favor of users, since they liberated users to remember only the existence of an important variable, but not its exact value at each moment.

These points become evident when comparing Guide-UI to SAW-UI. Since working memory is limited, users could not remember the exact time to rest (either for WBGT or noise) and what protective equipment they already had gathered.

Therefore, while using Guide-UI, they constantly forgot to rest and to equip or test several equipment (28 errors overall), but with SAW-UI they almost never missed to rest or forgot any equipment (4 errors overall), which explains some of the difference in Global SAW results when comparing UIs.

7.2.4.3 Workload, fatigue and other stressors

The solutions proposed for this problem are two: the Messaging mechanisms, especially the alarm and increased saliences, helps get the user attention when they are tired;

Personal Awareness agents can model stress levels and other agents can use this information for UI adaptation strategies.

Another proposed solution, but not explored in this thesis, is using gamification to motivate users [Oliveira15a].

In the experiments, neither the simulation execution order nor the test duration impacted user performance. Evaluators observed that the experts felt some level of fatigue in the final executions (because they performed the simulation eight times, 2 for training and 6 for evaluation). This factor, however, did not result in an increased number of mistakes. Figure 7.2 demonstrates this trend, showing the amount of errors committed in each simulation, which are ordered in the same sequence they were executed. Below each label the graphic shows the UI used: Guide-UI or SAW-UI.

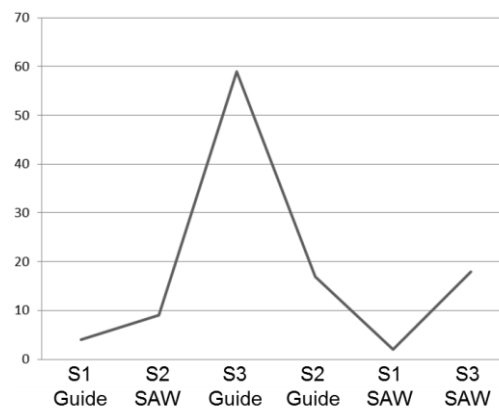


Figure 7.2: Number of errors committed in each situation simulated by order of execution. Errors committed with SAW-UI are always lower than those committed with Guide-UI.

Figure 7.2 shows that the difference in errors committed is clearly tied to the UI used and not to fatigue, since SAW-UI always led to fewer mistakes when comparing the same situation. For example, in situation S3, using SAW-UI led to $\frac{1}{3}$ of the errors compared to Guide-UI. S2 had a $\frac{1}{2}$ proportion, and S1, which was the least complex (and most similar to a daily routine) had a similar proportion between UIs. This fact also proves that the learning effect was not a problem in the experiment, since later situations had a consistent amount of errors.

7.2.4.4 Data overload

The two major approaches this thesis proposes for data overload is the decision-driven/goal-driven design and the SA model. The decision-driven/goal-driven design limits the information in the UI to the essential for the decision-making process, while the SA model

(and the layers of information) guarantees that only the information relevant for the current situation (and projected situations) is shown to users.

In the experiments, when asked if information was easy to locate, and if SAW-UI showed an ideal amount of information, experts responded positively, with respectively, a 1,2 and 1 average score in the Likert scale.

In addition, SART results for demand and supply demonstrated that data overload was not a problem.

7.2.4.5 Misplaced salience

The SA model, with the Messaging mechanism feature, allowed for salience to be increased when necessary in the right place.

However, when using SAW-UI “v1”, experts almost never rescued their coworker in S3, even when SAW-UI showed a map with someone asking for help. When asked the reason, they said they did not notice this information.

Two problems caused this: first the map was not relatable, no one could understand that it was a map and what exactly (or where) it was showing (the map was similar to the one in Figure 6.14 “Collective Safety”); second, the help sign was too discreet (even though it was increasing and decreasing in size).

The solution was iterating a new UI (SAW-UIv2, applied only for non-experts, as explained in Chapter 6) and to rework the map, flash colors on the help sign, and specially, open the Team Awareness interface which showed that the coworker was asking for help, and use the same alarm structure used for risks to call the user attention to this new information. It worked well, with all the non-experts rescuing their coworker (20% of experts made the rescue and they used SAW-UI v1, against 100% of non-experts and they used SAW-UI v2).

The easy process of modifying a UI, by just editing a Decision Component in an agent is what allowed for this fast improvement. Therefore the EUD approach of MBSAW-UI is the major answer to the “Misplaced salience” SAW demon.

7.2.4.6 Complexity creep

The MAS architecture, with each agent showing information related to an entity, divided the problem of acquiring SAW and diluted complexity (as discussed in Section 4.4.3 and Section 4.5.1).

Even with a wide range of events, users were able to comprehend their situations and perform well, as demonstrated in the efficacy results, showing a probable easiness in dealing with complexity when using SAW-UI.

Additionally, SART had a question about complexity of the situation and the answer comparing Guide-UI and SAW-UI (experts) and SAW-UI (non-experts) analyzed under ANOVA showed an equality on answers averages ($F < F_{critical}$; $1,1 < 3,7$; $P = 0,35$). This shows that the experience was similar to users' groups independent of UI, because the complexity was coming from the simulation itself, and not the UI.

7.2.4.7 Errant mental models

Fast and constant changes in the situation may cause errant mental models, because users do not notice and react to the change. For this problem, the SA model can be useful by the use of UI transitions and Messaging mechanisms (to create salience).

However, simple misinterpretation of the situation is still an open problem to solve. Data (and observation) shows that expert users disregarded some information in SAW-UI regarding sensors value. Since some of this information does not exist in real life (many of the sensors in SAW-UI), it was clear that they trusted their expertise and relied on their senses (especially vision) to decide whether or not the equipment was broken.

For instance, vibration in the slip ring was incorrectly reported 9 times by experts (5 times with Guide-UI and 4 with SAW-UI), and overheating of the slip ring was incorrectly reported 4 times by experts (1 time with Guide-UI and 3 with SAW-UI).

Although the confusion may have been caused due to low representation fidelity of these specific problems (or just a hard problem to identify), SAW-UI clearly showed contrary information when these errors were made.

Therefore, errant mental models and expert bias need to be addressed in future iterations of MBSAW-UI and future research.

7.2.4.8 Out-of-the-loop syndrome

FSA maps System Awareness (automation) as big part of SAW in industry, and the out-of-the-loop syndrome can be dealt by creating agents that support this awareness and that support decisions related to automation.

Nonetheless, automation was not explored in the study case to show statistical data that confirm the efficiency of this thesis towards automation. A future study case, with automation present, will be elaborated.

7.2.5 P2.C - SAOD guidelines

Proposition: the design process facilitates following the guidelines of SAW oriented development from the literature.

The SAOD guidelines was a set of guidelines created by Endsley and Jones [Endlsey12] to improve SASUI design and they were described in Section 2.3.2.2, and in this chapter some of them were already explored (Global SAW).

In Table 3.1 a list of features of this thesis to deal with these guidelines was presented., and now they will be presented and debated.

7.2.5.1 Organize information around goals.

By doing a GDTA, information is already grouped in goals during design. Nonetheless, MBSAW-UI reinforced this by using a decision-driven design approach, in which designers need to design UIs for each decision, which is the decision modeling process presented in Section 5.5.2.

Additionally, UIs still need to adapt at run-time for users' goals. The goal-based filtering promoted by the proposed architecture allows for agents to have triggers that activate their Decision Component and to cooperate with each other to warn of changes in the situation. This makes the UI aware of changes in the user goals when necessary to respond to the environment changes.

Considering the study case, some examples of using these strategies are: the Decision Component for WBG/Noise risk, modeled in the Section 6.4.2, is only activated when the values are enough to be considered a risk; the Messaging mechanism is used to switch procedures when an emergency is present.

7.2.5.2 Present Level 2 information directly

Initially, the GDTA list level 2 information. Then, there are two strategies for supporting level 2: using visual juxtaposition (positioning UI elements on top of each other) or calculating the condition of the entity and directly displaying it.

To support visual juxtaposition, designers can use the basic visual types, or even create their own UI component. Three examples in the study case of this guideline are: the UI component created in Section 5.3.3 (Figure 5.6) related information of past, present and future values of a sensor with its safety boundary, and this UI was used in Figure 6.25; in Figure 6.10, the UI associates information of two problems to make it clear that, combined, they represent a more risky situation; in Figure B-3 the SpecDC UI associates safety gear with their risk.

To support direct calculation of level 2 SAW, the SA model in the architecture allowed constructing a DFA of the situations of each agent and also defining a level 2 SAW dynamic equation. This calculation was then used in some cases to change the UI, such as in Figure 5.13, Figure 6.17 and Figure 6.18 and in others to show a direct value for users, such as in Figure 5.13, Figure 6.20.

7.2.5.3 Provide assistance for Level 3 SAW projections

The same strategies from level 2 SAW are applicable to level 3 SAW. Considering visual juxtaposition, the UI component created in Section 5.3.3 (Figure 5.6) related past and future values, assisting users in understanding the future value of the variable measured. Another example is on Figure 6.21, in which an arrow connecting information was created to guide users to projecting their future situation.

In case of the calculations, many Decision Components had a level 3 DFA that calculated the future state of the entity and changed the UI as a way to assist users in understanding this projected state, such as in Figure 6.17 and Figure 6.20. Also, most alarms in the system are based on level 3, so that users prepare for a new situation as soon as the current one starts to change.

7.2.5.4 Use information filtering carefully

Information filtering applied automatically by computers, like presenting information based on contexts, according to Endsley and Jones [Endsley12] may do more harm than good for three reasons: 1) the development of SAW is a process done overtime, and presenting information on demand may hinder this process; 2) it may deprive users of developing Global SAW and anticipating events; 3) different operators may use different information to develop their Situation Awareness, and computer-driven information filtering may damage this process for some users.

Nevertheless, this guideline was based on command, control and monitoring (C2) study cases, which differ in some aspects from maintenance work. For instance, while displays for C2 tasks are big, HMDs/Smartphones used for maintenance work have a very limited size and resolution to present information. Additionally, in C2, tasks are executed upon the display and the UI, while in maintenance work, tasks are executed in the physical world and UIs can only guide/assist users.

Streefkerk et al. recommended two strategies for information filtering [Streefkerk06]

- Limiting the choice between possible actions and disambiguating users' goals;
- Notify and request attention based on information priority.

The combination of these recommendations resulted in the strategy proposed in this thesis. Initially, the goal-based filtering serves to cut Decision Components that are not relevant. This strategy was combined with the SA model, which allowed for a level 2/3 filtering by permitting DFAs based on these levels to create UIs for each state, and use the Messaging mechanisms to transform UIs to request attention.

The second part of the proposed strategy is using the layers of information and customization capability to organize important information. With the layers of information, the Visible mode focused on providing Local Awareness during manual work, and the Nonvisible mode would update users Global Awareness during downtimes (or until the level 2/3 filtering triggered to classify an agent as a priority). These layers of information are associated with a customization approach that allowed users to freely change information from one layer to the other, so that they organized their UI based on their own needs.

These strategies, as discussed in Section 7.2.4.1 and Section 7.2.4.4, seemed to work.

7.2.5.5 Manage rampant featurism through prioritization and flexibility

The GDTA already listed only information that is important for the decision-making process. Nonetheless, two other strategies were used to manage featurism.

The partition dynamism in the UI prevented information of an Awareness to occupy more space than permitted. If too much information from an Awareness becomes important, a scrollbar is used to allow browsing information that is confined in the space of its Awareness.

The second strategy is the already discussed customization, which allows users to personalize their UI.

7.2.6 P2.D - SA model

Proposition: the Situation Assessment model, based on a multiagent architecture with autonomous SAW-driven interface agents, improve UIs support to SAW in the industry domain.

This is the most wide and abstract proposition, and to evaluate it the most high-level results will be showed. The overall results were that: experts liked and performed better using SAW-UI than Guide-UI; non-experts had a good experience with SAW-UI; performance results confirm the superiority of SAW-UI.

A proof of improved Situation Awareness provided by SAW-UI is the superiority regarding the error: “Did not coordinate with the team to execute another task when the equipment was being used”. This particular error is important because the only way to not commit it is to have reached a correct projection of the situation (level 3 of SAW).

Figure 7.3 demonstrates the user situation at the time of the interruption. Users still had 7 minutes of work before they had to stop the task because of the noise value, but at the same time they were being interrupted for 7 minutes before they could continue.

Using Guide-UI, 9 errors were committed by experts (90% of the occasions). Using SAW-UI, 1 error was committed by experts (10% of the occasions) and 1 by non-experts (16% of the occasions). The solution SAW-UI approached was explained in Section 6.4.2 and shown in Figure 6.21.

Finally, the answer to the question of providing an “ideal” Situation Awareness is not easy to obtain. SART-scores demonstrated a high evaluation of SAW-UI. Nonetheless, efficacy by users (non-experts) of SAW-UIv2 was the major parameter to achieving or not this “ideal” SAW. The higher the number of errors, the further from the ideal SAW would be.

Analyzing SAW-UI v2 results, 0,25 errors per simulation were committed regarding Local Awareness, 0,81 errors per simulation regarding Global Awareness and 0,58 errors per simulation were committed overall (total count). These results showed that less than one error is committed per simulation using the second iteration of SAW-U, and they are better than Guide-UI (2,7 errors per simulation overall) and SAW-UI v1 (0,7 errors per simulation overall).

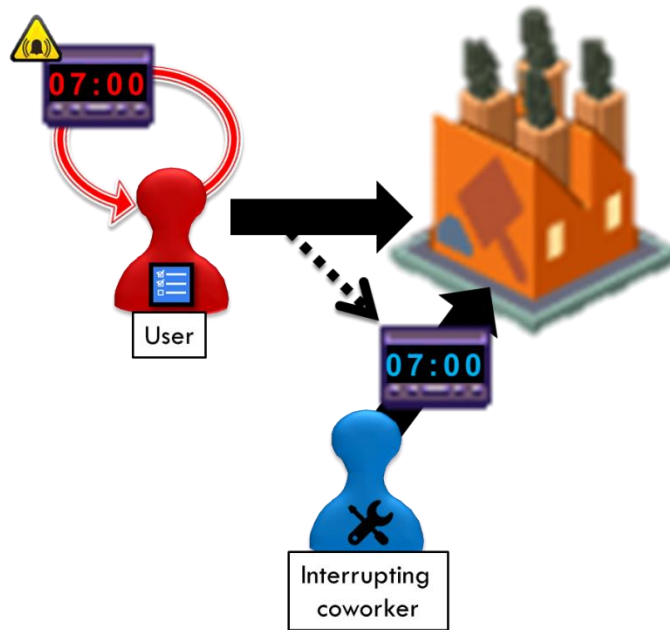


Figure 7.3: Illustration of the error: “Did not coordinate with the team to execute another task when the equipment was being used”.

7.2.7 P3 - applicability in the industry

Proposition: the model-based process develops products prepared for the industry domain.

To evaluate the validity of this thesis for the industry, three points will be discussed:

- Modeling capability: how malleable and flexible is the models that govern the system and its UI, so that end-users can increment the system;
- Context-awareness: does the system have context-awareness during its use;
- Acceptance: how is the acceptance of the system by industry workers.

7.2.8 P3.A - modeling capability

Proposition: The End-User Development approach allow the creation, by users, of new UIs for tasks, beside modifying existing ones, which supplies users with their own maintainability strategy for the system, leading to a reduced impact in the constant changes of the industry.

MBSAW-UI has two stages. The second stage, Model-Based SAW-oriented UI Design (Section 5.5) is the one when UIs are created and it is executed mainly by SMEs (with assistance from UI designers) in their own company.

Due to the model-based architecture, this thesis offers a solution that allows a system to be maintained by its own users. This means that maintainability, which is important in the industry domain, is one feature of MBSAW-UI. And not only is a feature, but it is also on the hands of the users.

One example of this feature is the creation of SAW-UIv2. After SAW-UIv1 was created and tests executed, there was a problem with a lack of Team Awareness, as discussed in Section 6.4.3. If the architecture of the system was not based on models, improving the UI would be a time consuming feature. But due to this model-based nature and the componentized structure, this improvement was made only by changing a Decision Component of one of the agents.

The discussion of agent reuse (Section 4.5.2) also offers insight on the flexibility and agility to model agents in the defined architecture.

Nonetheless, two open questions are: how much time does it take to train someone to use MBSAW-UI? How easy it is to learn and to use it to produce thoroughly SASUIs?

7.2.9 P3.B - adaptivity

Proposition: the final product (UI) has adaptivity to a wide range of situations.

This proposition was already debated and evaluated through some of the others, especially P2.C and Section 7.2.5.1.

7.2.10 P3.C - acceptability

Proposition: the final product (UI) is well-accepted by industry workers.

The conclusion of this proposition was that SAW-UI was well-accepted by experts from the industry. Two results can be analyzed to perceive industry acceptance of SAW-UI.

Usability of SAW-UI was well evaluated by experts and non-experts and component of Understanding within SART, which analyzes matters related to usability and easiness of UIs, had better results for SAW-UI. In fact, as shown in Figure 7.4, no expert gave a higher score for Guide-UI in the component of Understanding of SART.

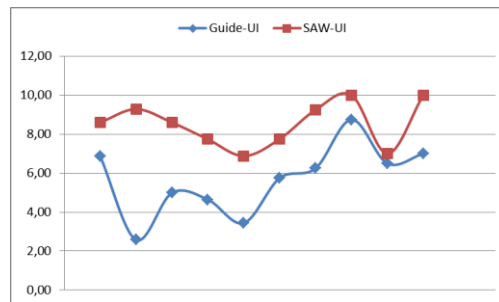


Figure 7.4: Guide-UI and SAW-UI comparison regarding the component of Understanding within the SART.

Finally, some subjective proofs of acceptance are the willingness experts had to collaborate (by testing) for free (no financial reward were given) and some suggestions written in the questionnaire to broaden the simulation both to cover more procedures in the inspection task, and to cover others tasks.

7.3 Summary of contributions

This chapter discussed the thesis propositions through an analysis of the study case results and of SAW quality indicators. Results from the experiment showed that SAW, efficacy and safety were very well evaluated in SAW-UI. The analysis discussed the items from Table 3.1 about the quality indicators for SAW support, and how both MBSAW-UI and the model-based MAS architecture led to a structured process for developing and designing Situation Awareness Support UIs (SASUI). Thus, a compilation of our contributions are:

- Model-based design methodology for SASUIs: a model-based design process was developed to facilitate the creation of Situation Awareness Support UIs for maintenance and industrial workers. The MBSAW-UI methodology gives a structure for developers and SMEs to gather and analyze requirements, establish SAW in the domain, and quickly start modeling, in a high-level, Users Interfaces. Later on, after the system is deployed, it allows for SMEs and UI designers to continue increasing the system coverage of tasks, by designing new UIs for new tasks, or updating older UIs to realign with tasks requirements;
- Framework for Situation Awareness Aspects (FSA): the proposed Cognitive Task Analysis for current SAOD (GDTA) does not have a representation technique. The FSA was initially made to cover this deficiency. It occasionally grew in importance, because it proposes a lateral division of SAW (the 7 Awareness: Personal, Procedure, Equipment,

Environment, System, Team and Enterprise), instead of only a vertical one (level 1, 2 and 3 of SAW).

- Autonomous SAW-driven interface agents: autonomous interface agents and SAW agents were different concepts. In this thesis both were aggregated, resulting in a new type of agent that, while still autonomous and responsible for User Interface generation, focuses on providing support for its users SAW processes. These agents are able to do so by being modeled in the form of decisions and not technology/functionalities. Additionally, through the Messaging mechanisms, agents can not only support user development of SAW, but also enforce their business rules;
- Model-based run-time environment architecture based on agents: this run-time architecture can be useful for developers of E-Maintenance systems. Some advantages of this architecture are:
 - Allows agents to easily access data from E-Maintenance (and derived) systems, which allows for designers to focus on a high-level problem (user SAW and necessary information);
 - It can work as a View (MVC) for E-Maintenance systems, facilitating the creation of UIs for these knowingly complex systems, because of its user-centered design strategy;
 - It understands that a big part of problems of current solutions is not giving users control over content creation. An End-User Development strategy was adopted to refer to this problem, allowing end-users (SME) to design (together with UI designers) their own UIs through MBSAW-UI, and the run-time environment is able to run it;
- Situation Awareness structural division in maintenance: the in-depth analysis of SAW in maintenance led to the establishment of its structural division (and to FSA);
- Local and Global SAW in maintenance: using the FSA, Local and Global SAW were more comprehensively defined for maintenance, which generate awareness for these terms (especially for Global SAW) and facilitate discussions of current UIs for this domain and their characteristics regarding SAW support;
- HUD-based User Interface for maintenance work: an innovative HUD-based User Interface was designed, in sight of current HMD devices, for the study case. Since the UI was designed for a real task and tested with workers used to execute this task, it can help

the industrial scene move forward by being an example and guideline for how should UIs for maintenance workers organize and project information regarding SAW;

- Study case and simulation: a simulation of a hydropower plant generator inspection in a virtual environment, with domain, task, cognitive and efficacy analysis, which could be used by the community to propose others UIs to solve this study case (and compare results);
- Some extra contributions are appendixes or published papers, such as the complete maintenance GDTA (Appendix A), the comprehensive version of the FSA [Oliveira14, Oliveira15b], a HUD and Augmented Reality high fidelity UI prototype for maintenance with gamification perspective [Oliveira15a].

7.4 Future works

A comprehensive list of possible future works derived from this thesis is:

- A complementary analysis on interruptions and reacquisition of SAW, as John and Smallman did [John08]. This is relevant because, although interruption is not an official SAW demon, it is a highly-regarded problem in the industrial domain [Carvalho15]. Its effects on SAW and a solution to deal with them are still an open problem.
- To test with large amount of agents (thousands), in which issues such as treatment of concurrent alarm triggering by agents cooperation will be addressed.
- Regarding the EUD paradigm, future works should improve on: the time it takes to train designers to use MBSAW-UI; Ensure that designers are consistent, modeling alarms only when really necessary; Ensure that different designers do not model the same exact decision (in different times);
- An abstraction layer can be created to extend MBSAW-UI, with Abstract UI elements that would adapt the UI based on context (devices, platform, etc);
- A negotiation strategy for the UI layout management, since currently information is displayed based on any agent interested in the situation, which may lead to information/screen overload. The master agent (layout manager) should ideally deal with this through a negotiation strategy, using auctions to negotiate what information is more valuable for the user at each time. These auctions could be implemented by a Contract Net Protocol using the first-price sealed bid auction technique [Bossé07], promoting a dispute

for a resource (space-time) in the UI. The layout manager could consider the following information to promote this negotiation strategy: user control (requisition); channel capacity (user mental capacity); transition time from acquisition to maintenance of SAW; usable space/occupation of UI; required space to render an agent UI; user expertise; weight, representativity and repetition of information; priority of supplying agent; current events; human factors and environmental variables;

- A strategy to deal with interruptions and reacquisition of SAW;
- An improvement to the support of processing and discovery of level 3 SAW (projection of situations);
- Implement a logic paradigm, allowing the description logic in the SA model to generate results that can be used to further infer information.
- UI adaptation at run-time to match users preferences regarding layout, language and end-device, and based on some Human Factors, such as experience, stress, motivation and cognitive capacities;
- Model and Control (MVC) aspects of System of Systems (E-Maintenance systems), including how to access (low level concerns) information from these systems and the Internet of Things;
- Dealing with data uncertainty in SAW modeling and visualization;
- Another study case for other activities/domains in the industrial scenario, which uses the System and Personal Awareness from the FSA model, to deal with automation and personalization respectively;

REFERENCES

- Ahrndt12a Ahrndt, S., Roscher, D., Lützenberger, M., Rieger, A., & Albayrak, S. (2012, March). Applying Model-Based Techniques to the Development of UIs for Agent Systems. In PAAMS (Workshops) (pp. 1-8).
- Ahrndt12b Ahrndt, S., Fähndrich, J., Lützenberger, M., Rieger, A., & Albayrak, S. (2012). An agent-based augmented reality demonstrator in the domestic energy domain. In *Advances on Practical Applications of Agents and Multi-Agent Systems* (pp. 225-228). Springer Berlin Heidelberg.
- Aleksy14 ALEKSY, Markus; RISSANEN, Mikko J. Utilizing wearable computing in industrial service applications. *Journal of Ambient Intelligence and Humanized Computing*, v. 5, n. 4, p. 443-454, 2014.
- Antonovsky14 Antonovsky, A. and Pollock, C. and Straker, L. 2014. Identification of the human factors contributing to maintenance failures in a petroleum operation. *Human Factors: The Journal of the Human Factors and Ergonomics Society*. 56 (2): pp. 306-321.
- Asai08 ASAI, Kikuo. The Role of Head-Up Display in Computer-Assisted Instruction. *Human Computer Interaction: New Developments*. Publisher: InTech, Chapter 2, pp. 33-48.
- Baader09 Baader, F., Bauer, A., Baumgartner, P., Cregan, A., Gabaldon, A., Ji, K., ... & Schwitter, R. (2009). A novel architecture for situation awareness systems. In *Automated Reasoning with Analytic Tableaux and Related Methods* (pp. 77-92). Springer Berlin Heidelberg.
- Baines05 Baines T., Asch R., Hadfield L., Mason J., Fletcher S., Kay J. —Towards a Theoretical Framework for Human Performance Modeling Within Manufacturing Systems Design. *Simulation Modeling Practice and Theory*, 13, 2005, pp. 486–504.
- Balland13 Balland, E., Consel, C., N'Kaoua, B., & Sauzéon, H. (2013, May). A case for human-driven software development. In *Proceedings of the 2013 International Conference on Software Engineering* (pp. 1229-1232). IEEE Press.
- Bangemann06 Bangemann, T., X. Rebeuf, D. Reboul, A. Schulze, J. Szymanski, J-P Thomesse, M. Thron, and N. Zerhouni. PROTEUS - Creating Distributed Maintenance Systems through an Integration Platform. *Journal of Computers in Industry*, 2006; 57 (6): 539–551.

- Benbelkacem11 Benbelkacem S., Zenati-Henda N., Zerarga F., Bellarbi A., Belhocine M., Malek S., Tadjine M., Augmented Reality Platform for Collaborative E-Maintenance Systems, InTech, 2011, 211-226.
- Benyon14 BENYON, David. Designing Interactive Systems: A comprehensive guide to HCI, UX and interaction design, Person, 3/E, 2014.
- Bernier01 Bernier, Y. W. (2001, March). Latency compensating methods in client/server in-game protocol design and optimization. In Game Developers Conference (Vol. 98033, No. 425).
- Blasch12 Blasch, E., Valin, P., Joussemme, A. L., Lambert, D., & Bossé, É. (2012, July). Top ten trends in high-level information fusion. In Information Fusion (FUSION), 2012 15th International Conference on (pp. 2323-2330). IEEE.
- Blom15 BLOM, Henk AP; SHARPANSKYKH, Alexei. Modelling situation awareness relations in a multiagent system. Applied Intelligence, v. 43, n. 2, p. 412-423, 2015.
- Bouyer05 BOUYER, G. C.; SZNELWAR, L. I. Análise Cognitiva do Processo de Trabalho em Sistemas Complexos de Operações. [Cognitive analysis of the work process in Complex Systems of Operations]. Ciências & Cognição (UFRJ) [Science & Cognition], v. 4, n.2, p. 2-24, 2005. (In portuguese).
- Bolstad06 BOLSTAD, C. A.; COSTELLO, A. M.; ENDSLEY, M. R. Bad situation awareness designs: What went wrong and why. In: Proceedings of the 16th World Congress of International Ergonomics Association. 2006.
- Bossé07 BOSSÉ, E.; ROY, Jean; WARK, Steve. Concepts, Models, and Tools for Information Fusion, Artech House. Inc, 2007.
- Bosse12 BOSSE, Tibor; MERK, Robbert-Jan; TREUR, Jan. Modelling temporal aspects of situation awareness. In: Neural Information Processing. Springer Berlin Heidelberg, 2012. p. 473-483.
- Bosse13 Bosse, T., Majdanik, K., Boersma, K., & Ingibergsdóttir, K. Studying Shared Situation Awareness by Agent-Based Simulation. In: Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 02. IEEE Computer Society, 2013. p. 201-208.
- Botega15 Botega, L. C., Ferreira, L. C., Oliveira, N. P., Oliveira, A., Berti, C. B., de Neris, V. P., & de Araújo, R. B. (2015). SAW-oriented user interfaces for emergency dispatch systems. In Human Interface and the Management of Information. Information and Knowledge in Context (pp. 537-548). Springer International Publishing.

- Botega16 Botega L., "Modelo De Fusão Dirigido Por Humanos E Ciente De Qualidade De Informação". [Human Driven and Quality Aware Model Fusion]. 2016. Thesis (Federal University of São Carlos). (In Portuguese).
- Bullemer13 Bullemer, P., Reising, D.: Improving the Operations Team Situation Awareness: Lessons Learned from Major Process Industry Incidents. In: American Fuel & Petrochemical Manufacturers Annual Meeting, pp. 1–12 (2013)
- Campbell14 Campbell, A. G., Stafford, J. W., Holz, T., & O'Hare, G. M. (2014). Why, when and how to use augmented reality agents (AuRAs). *Virtual Reality*, 18(2), 139-159.
- Carroll03 Carroll, J.M., Neale, D.C., Isenhour, P.L., Rosson, M.B., ScottMcCrickard, D.: Notification and awareness: synchronizing task-oriented collaborative activity. *International Journal of Human-Computer Studies* 58(5), 605–632 (2003)
- Carvalho15 CARVALHO, Alex Luis de; MENEGON, Nilton Luiz. The invisible, dangerous work of maintenance professionals: reflections on activity in the automotive industry. *Production*, v. 25, n. 1, p. 201-202, 2015. (In Portuguese).
- Chinoy11 Chinoy, S., Fischer, D.: Advancing Situational Awareness for Technical Operations. In: Annual Air Traffic Control Association Conference Proceedings, pp. 16–27 (2011)
- Crandall13 Crandall, Beth W., and Robert R. Hoffman. (2013). *Cognitive Task Analysis*. The Oxford handbook of cognitive engineering.
- Dekker04 Dekker, S. W. A., & Hollnagel, E. (2004). Human factors and folk models. *Cognition, Technology, and Work*, 6, 79–86.
- Dhillon06 Dhillon, B., Liu, Y.: Human error in maintenance: a review. *Journal of Quality in Maintenance Engineering* 12(1), 21–36 (2006)
- Drey09 Drey, Z., Mercadal, J. and Consel, C., 2009, January. A taxonomy-driven approach to visually prototyping pervasive computing applications. In *Domain-Specific Languages* (pp. 78-99). Springer Berlin Heidelberg.
- Drey10 DREY, Zoé; CONSEL, Charles. A visual, open-ended approach to prototyping ubiquitous computing applications. In: *Pervasive Computing and Communications Workshops (PERCOM Workshops)*, 2010 8th IEEE International Conference on. IEEE, 2010. p. 817-819.
- Drey12 DREY, Zoé; CONSEL, Charles. Taxonomy-driven prototyping of home automation applications: A novice-programmer visual language and its evaluation. *Journal of Visual Languages & Computing*, v. 23, n. 6, p. 311-326, 2012.

- Dubois14 Dubois, E., Bortolaso, C., Appert, D., & Gauffre, G. (2014). An MDE-based framework to support the development of Mixed Interactive Systems. *Science of Computer Programming*, 89, 199-221.
- Durso06 DURSO, F. T.; BLECKLEY, M. K.; DATTEL, A. R. Does Situation Awareness Add to the Validity of Cognitive Tests? *Human Factors*, [S.l.], v.48, n.4, p.721–733, 2006.
- Eisenstein02 EISENSTEIN, Jacob; RICH, Charles. Agents and GUIs from task models. In: *Proceedings of the 7th international conference on Intelligent user interfaces*. ACM, 2002. p. 47-54.
- Enblom15 ENBLOM, Gustav; ESKEBAEK, Hannes. Real Time Vehicle Diagnostics Using Head Mounted Displays. 2015. Thesis (Linköpings universitet)
- Endsley89 Endsley, M.R., 1989. Final report: situation awareness in an advanced strategic mission. NOR DOC 89-32, Northrop Corporation, Hawthorne, CA.
- Endsley95a ENDSLEY, M.R., 1995a, Towards a theory of situation awareness in dynamic systems. *Human Factors*, 37, pp. 32–64.
- Endsley95b ENDSLEY, Mica R. Measurement of situation awareness in dynamic systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, v. 37, n. 1, p. 65-84, 1995.
- Endsley00a Endsley, M. R., & Robertson, M. M. (2000a). Situation awareness in aircraft maintenance teams. *International Journal of Industrial Ergonomics*, 26, 301-325.
- Endsley00b ENDSLEY, Mica R. Direct measurement of situation awareness: Validity and use of SAGAT. *Situation awareness analysis and measurement*, v. 10, 2000.
- Endsley12 Endsley, M. R., & Jones, D. G. (2012). *Designing for situation awareness: An approach to human-centered design* (2nd ed.). London: Taylor & Francis.
- Endsley13 ENDSLEY, Mica R. Situation awareness-oriented design. *The Oxford Handbook of Cognitive Engineering*, p. 272, 2013.
- Endsley15a ENDSLEY, M.R., 2015, Situation Awareness Misconceptions and Misunderstandings. *Journal of Cognitive Engineering and Decision Making* March 1, 2015, 9, pp. 4–32.
- Endsley15b ENDSLEY, Mica R. Situation awareness: operationally necessary and scientifically grounded. *Cognition, Technology & Work*, v. 17, n. 2, p. 163-167, 2015.

- Espindola11 D. Espíndola, L. Fumagalli, M. Garetti, S. Botelho, C. Pereira, An Adaption of OSACBM architecture for human-computer interaction through mixed interface, in: INDIN 11 Conference, Caparica, Lisbon, Portugal, 26–29, 2011.
- Espindola13 Espíndola D., Fumagalli L., Garetti M., Pereira C., Botelho S., Henriques R, “A model-based approach for data integration to improve maintenance management by mixed reality”, *Comput. Ind.*, vol. 64, 2013, pp. 376-391.
- Fan05 X. Fan, S. Sun, M. McNeese, & J. Yen, Extending Recognition-Primed Decision Model For Human-Agent Collaboration, In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi Agent Systems*, pp. 945-952, The Netherlands, July 25-29, 2005
- Feiner93 FEINER, Steven; MACINTYRE, Blair; SELIGMANN, Dorée. Knowledge-based augmented reality. *Communications of the ACM*, v. 36, n. 7, p. 53-62, 1993.
- Feng09 FENG, Yu-Hong; TENG, Teck-Hou; TAN, Ah-Hwee. Modelling situation awareness for Context-aware Decision Support. *Expert Systems with Applications*, v. 36, n. 1, p. 455-463, 2009.
- Fiorentino16 Fiorentino, M., Uva, A. E., Monno, G., & Radkowski, R. (2016). Natural interaction for online documentation in industrial maintenance. *International Journal of Computer Aided Engineering and Technology*, 8(1-2), 56-79.
- FIPA The Foundation for Intelligent Physical Agents – see www.fipa.org
- Friedrich02 Friedrich W., ARVIKA-augmented reality for development, production and service, *Proc. ISMAR 2002*, 2002, 3-4.
- Gersh05 Gersh, John R., McKneely, Jennifer A. and Remington, Roger W. (2005) Cognitive engineering: Understanding human interaction with complex systems. *Johns Hopkins APL Technical Digest*, 26 4: 377-382.
- Golightly10 Golightly, D., Wilson, J. R., Lowe, E., & Sharples, S. (2010). The role of situation awareness for understanding signalling and control in rail operations. *Theoretical Issues in Ergonomics Science*, 11(1-2), 84-98.
- Golightly13 Golightly D., Ryan B., Dadashi N., Pickup L., Wilson J.R., “Use of scenarios and function analyses to understand the impact of situation awareness on safe and effective work on rail tracks”, *Safety Science*, vol. 56, 2013, pp. 52-62.
- Greenberg94 Greenberg, S. and D. Marwood (1994): Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface. *Proceedings of the Conference on Computer-Supported Cooperative Work*. Chapel Hill NC, pp. 207–217.

- Gram97 Gram, C. and Cockton, G., editors (1997). Design principles for interactive software. Springer, London, UK. ISBN: 0412724707.
- Gutwin02 Carl Gutwin and Saul Greenberg. 2002. A Descriptive Framework of Workspace Awareness for Real-Time Groupware. *Journal Computer Supported Cooperative Work* 11, 3 (November 2002), 411-446.
- Halme97 Halme A., Visala A., Zhang X., Process Modeling Using the Functional State Approach Multiple model approaches to nonlinear modelling and control. CRC press, 1997.
- Helander06 Helander M., "Noise and Vibration", A Guide to Human Factors and Ergonomics, 2nd edn. CRC Press, 2006.
- Henderson10 Henderson, S., Feiner, S.: Exploring the Benefits of Augmented Reality Documentation for Maintenance and Repair. *IEEE Transactions on Visualization and Computer Graphics* 16(1), 4–16 (2010)
- Hoogendoorn11a Hoogendoorn, Mark, Rianne M. Van Lambalgen, and Jan Treur. An integrated agent model addressing situation awareness and functional state in decision making. In: *Agents in Principle, Agents in Practice*. Springer Berlin Heidelberg, 2011. p. 385-397.
- Hoogendoorn11b Hoogendoorn, Mark, Rianne M. Van Lambalgen, and Jan Treur. "Modeling situation awareness in human-like agents using mental models." In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, no. 1, p. 1697. 2011.
- Irizarry13 Irizarry, J., Gheisari, M., Williams, G., & Walker, B. N. (2013). InfoSPOT: A mobile Augmented Reality method for accessing building information through a situation awareness approach. *Automation in Construction*, 33, 11-23.
- Jantunen11 Jantunen E., Emmanouilidis C., Arnaiz A., Gilabert E., E-Maintenance: trends, challenges and opportunities for modern industry, Proc. of the 18th IFAC World Congress, 2011, 453-458.
- Jennings99 Jennings, N. R. Agent-oriented software engineering. In: *Proceedings of the 9th European Workshop on Modeling Autonomous Agent In a Multi-agent System Engineering (MAAMAW-99)*, Heidelberg, Germany, Springer-Verlag, pp.1–7, 1999.
- John08 John M., Smallman H., "Staying Up to Speed: Four Design Principles for Maintaining and Recovering Situation Awareness", *Journal of Cognitive Engineering and Decision Making*, vol. 2, 2008, pp. 118-139.
- Jones09 JONES, Rashaad ET; CONNORS, Erik S.; ENDSLEY, Mica R. Incorporating the human analyst into the data fusion process by modeling

- situation awareness using fuzzy cognitive maps. In: Information Fusion, 2009. FUSION'09. 12th International Conference on. IEEE, 2009. p. 1265-1271.
- Jones10 Modeling situation awareness for Army infantry platoon leaders using fuzzy cognitive mapping techniques. In: Proceedings of the Behavior Representation in Modeling and Simulation (BRIMS) Conference. 2010. p. 216-223.
- Jones11a Jones, Rashaad ET, Erik S. Connors, and Mica R. Endsley. "A framework for representing agent and human situation awareness." In Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), 2011 IEEE First International Multi-Disciplinary Conference on, pp. 226-233. IEEE, 2011.
- Jones11b Jones, R. E., Connors, E. S., Mossey, M. E., Hyatt, J. R., Hansen, N. J., & Endsley, M. R. Using fuzzy cognitive mapping techniques to model situation awareness for army infantry platoon leaders. Computational and Mathematical Organization Theory, v. 17, n. 3, p. 272-295, 2011.
- Jones15 JONES, Rashaad ET. Artificial Intelligence and Human Teams Examining the Role of Fuzzy Cognitive Maps to Support Team Decision-Making in a Crisis-Management Simulation. In: Proceedings of the Human Factors and Ergonomics Society Annual Meeting. SAGE Publications, 2015. p. 190-194.
- Kabac15 KABÁČ, Milan; CONSEL, Charles. Orchestrating masses of sensors: a design-driven development Approach. In: Proceedings of the 2015 ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences. ACM, 2015. pp. 117-120.
- Kay90 Kay, A. (1990) User interface: a personal view. In Laurel, B. (ed.), The Art of Human-Computer Interface Design. Addison Wesley, Reading, MA.
- Klein93 KLEIN, Gary. Naturalistic decision making: Implications for design. KLEIN ASSOCIATES INC FAIRBORN OH, 1993.
- Kokar12 KOKAR, Mieczyslaw M.; ENDSLEY, Mica R. Situation awareness and cognitive modeling. IEEE Intelligent Systems, n. 3, p. 91-96, 2012.
- Kriouile13 KRIOUILE, Abdelouahed; GADI, Taoufiq; BALOUKI, Youssef. CIM to PIM Transformation: A criteria Based Evaluation. International Journal of Computer Technology and Applications, v. 4, n. 4, p. 616, 2013.
- Kruchten95 KRUNTCHEN, P. Architectural blueprints—the” 4+ 1” view model of software architecture. IEEE Software, v. 12, n. 6, p. 42-50, 1995.

- Lieberman97 Lieberman, Henry. "Autonomous interface agents." In Proceedings of the ACM SIGCHI Conference on Human factors in computing systems, pp. 67-74. ACM, 1997.
- Lieberman06 Lieberman H, Paternò F, Klann M, Wulf V. End-user development: An emerging paradigm. Springer Netherlands; 2006.
- Lipson98 Lipson H., Shpitalni M., Kimura F., Gon-charenko I., Online Product Maintenance by Web-Based Augmented Reality, Proc. of New Tools and Workflows for Product Development, 1998, 131-143.
- Liu12 Liu, C., Exploring Mobile Augmented Reality Instructions to Assist Operating Physical Interfaces, Thesis (RWTH Aachen University)
- Livingston11 Livingston, M. A., Ai, Z., Karsch, K., & Gibson, G. O. (2011). User interface design for military AR applications. *Virtual Reality*, 15(2-3), 175-184.
- Loia16 LOIA, Vincenzo et al. Enforcing situation awareness with granular computing: a systematic overview and new perspectives. *Granular Computing*, p. 1-17, 2016.
- Lu08 LU, Jie; YANG, Xiaowei; ZHANG, Guangquan. Support vector machine-based multi-source multi-attribute information integration for situation assessment. *Expert Systems with Applications*, v. 34, n. 2, p. 1333-1340, 2008.
- Lynas11 Lynas, D., Horberry, T.: Human factor issues with automated mining equipment. *The Ergonomics Open Journal* 4(S2-M3), 74–80 (2011)
- marketsandmarkets.com14 marketsandmarkets.com, Head Mounted Display (HMD) Market by Products (Helmet Mounted, Wearable Glass), Components (Micro display, Camera, control unit, Tracker, Accessories), Applications (Defence, industrial, Video Gaming) & Geography - Global Analysis and Forecast to 2020. 2014.
- Matheus03 MATHEUS, Christopher J.; KOKAR, Mieczyslaw M.; BACLAWSKI, Kenneth. A core ontology for situation awareness. In: Proceedings of the Sixth International Conference on Information Fusion. 2003. p. 545-552.
- Matthews02 MATTHEWS, M.; BEAL, S. A. Assessing Situation Awareness in Field Training Exercises. [S.l.]: U.S. Army Research Institute for the Behavioural and Social Sciences, 2002.
- Meixner14 MEIXNER, Gerritt; CALVARY, Gaelle; COUTAZ, Joelle. Introduction to Model-Based User Interfaces. *W3C*, January, v. 7, 2014.
- Mercadal13 MERCADAL, Julien; DREY, Zoé; CONSEL, Charles. Denotational Semantics of A User-Oriented, Domain-Specific Language. arXiv preprint arXiv:1309.5141, 2013.

- Militello13 Militello, L. G., & Klein, G. (2013). Decision-centered design. *The Oxford handbook of cognitive engineering*, 261-271.
- Miller56 Miller, George A., The magical number seven, plus or minus two: Some limits on our capacity for processing information, *Psychological Review*, 1956, Vol 63, 81–97
- Miller99 Miller R., Griffin M., Hart P., —Personality and organizational health: The role of conscientiousness, *Work Stress* 13, 1999, pp. 7–19.
- Moreno-Trejo13 MORENO-TREJO, Jorge; HERAS, Enrique; MARKESET, Tore. Using the HAZOP analysis in the maintenance, modifications and installation of subsea petroleum production systems. In: *Reliability and Maintainability Symposium (RAMS), 2013 Proceedings-Annual*. IEEE, 2013. p. 1-6.
- Mostafa13 Mostafa, S., Ahmad, M.S., Ahmad, A. and Annamalai, M., 2013, December. Formulating Situation Awareness for Multi-agent Systems. In *Advanced Computer Science Applications and Technologies (ACSAT), 2013 International Conference on* (pp. 48-53). IEEE.
- Mostafa14 Mostafa, S. A., Ahmad, M. S., Tang, A. Y., Ahmad, A., Annamalai, M., & Mustapha, A. (2014). Agent's Autonomy Adjustment via Situation Awareness. In *Intelligent Information and Database Systems* (pp. 443-453). Springer International Publishing.
- Myers00 Myers, B.; Rosson, M. B.: Survey on User Interface Programming. *Proc. of the 10th Annual CHI Conference on Human Factors in Computing Systems*, pp. 195-202, 2000.
- Naderpour12 NADERPOUR, Mohsen; LU, Jie. A fuzzy dual expert system for managing situation awareness in a safety supervisory system. In: *Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on*. IEEE, 2012. p. 1-7.
- Naderpour14 NADERPOUR, Mohsen; LU, Jie; ZHANG, Guangquan. An intelligent situation awareness support system for safety-critical environments. *Decision Support Systems*, v. 59, p. 325-340, 2014.
- Naderpour15 NADERPOUR, Mohsen; NAZIR, Salman; LU, Jie. The role of situation awareness in accidents of large-scale technological systems. *Process Safety and Environmental Protection*, v. 97, p. 13-24, 2015.
- Nazir12 Nazir, S., Colombo, S., Manca, D.: The role of Situation Awareness for the Operators of Process Industry. *Chemical Engineering Transactions* 26, 303–308 (2012)
- Neisser76 Neisser, U. (1976): *Cognition and Reality*. San Fransisco: W.H. Freeman.

- Nicolai06 Nicolai, T., Sindt, T., Witt, H., Reimerdes, J., & Kenn, H. (2006, March). Wearable computing for aircraft maintenance: Simplifying the user interface. In *Applied Wearable Computing (IFAWC), 2006 3rd International Forum on* (pp. 1-12). VDE.
- Nielsen93 Nielsen J. "Usability Engineering", Morgan Kaufmann Publishers Inc., USA. 1993.
- Nwiabu12 N. Nwiabu, I. Allison, P. Holt, P. Lowit, and B. Oyeneyin. User interface design for situation-aware decision support systems, 2012 IEEE Int. Multi- Disciplinary Conf. Cogn. Methods Situat. Aware. Decis. Support, pp. 332–339, Mar. 2012.
- Oedewald03 Oedewald, Pia, and Teemu Reiman. "Core task modelling in cultural assessment: A case study in nuclear power plant maintenance." *Cognition, Technology & Work* 5.4 (2003): 283-293.
- Oliveira13 OLIVEIRA, A.; ARAUJO, R.; JARDINE, A. A Human Centered View on E-Maintenance. *Chemical Engineering Transactions*, v. 33, p. 385-390, 2013.
- Oliveira14 Oliveira, A., Araujo, R., & Jardine, A. (2014, June). Human-Centered Interfaces for Situation Awareness in Maintenance. In *International Conference on Human Interface and the Management of Information* (pp. 193-204). Springer International Publishing.
- Oliveira15a Oliveira, A., Caetano, N., Botega, L. C., & de Araújo, R. B. (2015). A Head-up Display with Augmented Reality and Gamification for an E-Maintenance System: Using Interfaces and Gamification to Motivate Workers in Procedural Tasks. In *Human Interface and the Management of Information. Information and Knowledge in Context* (pp. 499-510). Springer International Publishing.
- Oliveira15b Oliveira, A. C. M., Van Volkenburg, C. A., Jardine, A. K., & de Araujo, R. B. (2015, January). The electronic maintenance situational awareness interface. In *2015 Annual Reliability and Maintainability Symposium (RAMS)* (pp. 1-6). IEEE.
- OMAGame Services Architecture 11
Open Mobile Alliance, Game Services Architecture, 2011. Available at: http://technical.openmobilealliance.org/Technical/Release_Program/docs/GS-CSI/V1_0-20110329-A/OMA-AD-Game-Services-Architecture-V1_0-20110329-A.pdf
- Onal13 E. Onal, C. Craddock, M. R. Endsley, and A. Chapman, "From theory to practice: How designing for situation awareness can transform confusing, overloaded shovel operator interfaces, reduce costs, and increase safety". *ISARC 2013*, pp. 1517-1525

- Onwubiko12 ONWUBIKO, Cyril. Modelling Situation Awareness Information and System Requirements for the Mission using Goal-Oriented Task Analysis Approach. *Situational Awareness in Computer Network Defense: Principles, Methods and Applications*, p. 245, 2012.
- Opto2213 Opto 22. 2013. 'Building an HMI that Works: New Best Practices for Operator Interface Design'.
http://www.opto22.com/documents/2061_High_Performance_HMI_white_paper.pdf
- Paelke15 PAELKE, Volker et al. User interfaces for cyber-physical systems. *at-Automatisierungstechnik*, v. 63, n. 10, p. 833-843, 2015.
- Parasuraman08 Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (2008). Situation awareness, mental workload and trust in automation: Viable empirically supported cognitive engineering constructs. *Journal of Cognitive Engineering and Decision Making*, 2(2), 140-160.
- Parsons03 Parsons K., *Human Thermal Environments: The effects of hot, moderate, and cold environments on human health, comfort and performance*, 2nd edn. CRC Press, 2003.
- Paterno01 Paternò, F. (2001). *Model-based Design and Evaluation of Interactive Applications*, Springer Verlag.
- Paterno13 Paternò, F., 2013. *End user development: Survey of an emerging field for empowering people*. ISRN Software Engineering, 2013.
- Patrick10 PATRICK, John; MORGAN, Philip L. Approaches to understanding, analysing and developing situation awareness. *Theoretical Issues in Ergonomics Science*, v. 11, n. 1-2, p. 41-57, 2010.
- Pritchett00 Pritchett, A. R., & Hansman, R. J. (2000). Use of testable responses for performance-based measurement of situation awareness. In M. R. Endsley & D. J. Garland (Eds.), *Situation awareness analysis and measurement*. Mahwah, NJ: Lawrence Erlbaum.
- Rasmussen82 Rasmussen, J. (1982). Human errors. A taxonomy for describing human malfunction in industrial installations. *Journal of Occupational Accidents*, 4(2-4), 311-333.
- Ricci15 RICCI, Alessandro et al. *Programming Mirror Worlds: An Agent-Oriented Programming Perspective*. Revised, Selected, and Invited Papers of the Third International Workshop on Engineering Multi-Agent Systems, 2015.
- Robertson98 Michelle Robertson, "Maintenance Resource Management", *Human Factors in Aviation Maintenance and Inspection (HFAMI): Ten Years of Research and Development*, Federal Aviation Administration, 1998.

- Romero-Gómez13 Romero-Gómez, R., Tena, S., Díez, D., & Díaz, P. (2013). The Application of Situation Awareness-Oriented Design to the Smart Grid Domain. XIV Congreso Internacional de Interacción Persona-Ordenado. 2013.
- Roscher11 Roscher, D., Lehmann, G., Schwartz, V., Blumendorf, M., & Albayrak, S. (2011). Dynamic distribution and layouting of model-based user interfaces in smart environments. In *Model-Driven Development of Advanced User Interfaces* (pp. 171-197). Springer Berlin Heidelberg.
- Russel95 Russell, S. and Norvig, P. (1995) *Artificial Intelligence: a Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ.
- Salmon07 SALMON, P.; STANTONA, N.; WALKERA, G.; DAMIAN, G. Situation awareness measurement: a review of applicability for c4i environments. *Applied Ergonomics*, [S.l.], v.38, n.1, 2007.
- Schiaffino04 SCHIAFFINO, Silvia; AMANDI, Analía. User-interface agent interaction: personalization issues. *International Journal of Human-Computer Studies*, v. 60, n. 1, p. 129-148, 2004.
- Shehory98 Shehory, O. *Architectural properties of multiagent systems*. Pittsburgh, PA: [s.n.], 1998.
- Sian96 Sian, Benjamin, Michelle Robertson, and Jean Watson. "Maintenance resource management handbook." Federal Aviation Administration Office of Aviation Medicine, Washington, DC (1996).
- Sidney10 Sidney, W. D., Nyce, J. M., van Winsen, R., & Henriqson, E. (2010). Epistemological self-confidence in human factors research. *Journal of Cognitive Engineering and Decision Making*, 4(1), 27-38.
- Smolensky SMOLENSKY, M. W. Toward the physiological measurement of situation awareness: The case for eye movement measurements. In: *Proceedings of the Human Factors and Ergonomics Society 37th annual meeting*. 1993.
- Sneddon13 Sneddon, A., Mearns, K., Flin, R.: Stress, fatigue, situation awareness and safety in offshore drilling crews. *Safety Science* 56, 80–88 (2013)
- So04 So, R., & Sonenberg, L. (2004). Situation awareness in intelligent agents: Foundations for a theory of proactive agent behavior. In *Proceedings of IEEE/WIC/ACM international conference on intelligent agent technology* (pp. 86–92).
- Stanton06 Stanton, N.A., Stewart, R., Harris, D., Houghton, R.J., Baber, C., McMaster, R., Salmon, P., Hoyle, G., Walker, G.H., Young, M.S., Linsell, M., Dymott, R. and Green, D.A., 2006, *Distributed situation awareness in dynamic systems: theoretical development and application of an ergonomics methodology*. *Ergonomics*, 45, pp. 1288–1311.

- Stanton13 Stanton, N. A., Salmon, P. M., Walker, G. H., Baber, C. & Jenkins, D. P. (2013). *Human Factors Methods: A Practical Guide for Engineering and Design*. Ashgate Publishing, Ltd.
- Stanton15 STANTON, Neville A.; SALMON, Paul M.; WALKER, Guy H. Let the reader decide a paradigm shift for situation awareness in sociotechnical systems. *Journal of Cognitive Engineering and Decision Making* 2015, Volume 9, Number 1, March 2015, pp. 44–50.
- Starr10 Starr A., Al-Najjar B., Holmberg K., Jantunen E., Bellew J., Albarbar A., 2010, *Maintenance Today and Future Trends*, In *E-maintenance*, Springer-Verlag London Limited, 5-37.
- Stewart08 Stewart, R., Stanton, N. A., Harris, D., Baber, C., Salmon, P., Mock, M., Tatlock, K., Wells, L., Kay, A. (2008). Distributed situation awareness in an Airborne Warning and Control System: application of novel ergonomics methodology. *Cognition, Technology & Work*, 10(3), 221-229.
- Stock05 Stock I., Weber M., Steinmeier E., Metadata based authoring for technical documentation, *Proc. SIGDOC 2005*, 2005, 60-67.
- Streefkerk06 STREEFKERK, Jan Willem; VAN ESCH-BUSSEMAKERS, Myra P.; NEERINCX, Mark A. Designing personal attentive user interfaces in the mobile public safety domain. *Computers in Human Behavior*, v. 22, n. 4, p. 749-770, 2006.
- Taylor90 TAYLOR, R. M. Situational Awareness Rating Technique(SART): The development of a tool for aircrew systems design. *AGARD, Situational Awareness in Aerospace Operations* 17 p(SEE N 90-28972 23-53), 1990.
- Tran09 Tran, V., Kolp, M., Vanderdonckt, J., Wautelet, Y., & Faulkner, S. (2009). Agent-based user interface generation from combined task, context and domain models. In *Task Models and Diagrams for User Interface Design* (pp. 146-161). Springer Berlin Heidelberg.
- Tretten11 TRETTEEN, Phillip; KARIM, Ramin; KUMAR, Uday. Usability-based eMaintenance for effective performance measurement. In: *MPMM 2011: Maintenance Performance Measurement & Management: Conference Proceedings*. Luleå: Luleå tekniska universitet, 2011. p. 53-59.
- Tretten14 TRETTEEN, Phillip; KARIM, Ramin. Enhancing the usability of maintenance data management systems. *Journal of Quality in Maintenance Engineering*, v. 20, n. 3, p. 290-303, 2014.
- Truyen06 TRUYEN, Frank. *The Fast Guide to Model Driven Architecture The Basics of Model Driven Architecture*. Cephass Consulting Corp, 2006.

- United Kingdom Air Accidents Investigation Branch. (1990). Report on the accident to Boeing 737-400 G-OBME near Kegworth, Leicestershire on 8 January 1989 (Report No. 4/90 (EW/C1095)). Farnborough, U.K.: Department for Transport.
- Urlings03 Urlings, P., Tweedale, J., Sioutis, C., & Ichalkaranje, N. (2003). Intelligent agents and situation awareness. In Proceedings of the 7th international conference on knowledge-based intelligent information and engineering systems, LNCS 2774 (pp. 723–733).
- Vercellis09 Vercellis, C. Business intelligence: data mining and optimization for decision making. Editorial John Wiley and Sons, 2009.
- Viano00 Viano, G., Parodi, A., Alty, J., Khalil, C., Angulo, I., Biglino, D., ... & Lachaud, P. (2000, May). Adaptive user interface for process control based on multi-agent approach. In Proceedings of the working conference on Advanced visual interfaces (pp. 201-204). ACM.
- Villaren12 VILLAREN, Thomas; COPPIN, Gilles; LEAL, Angélica. Modeling task transitions to help designing for better situation awareness. In: Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems. ACM, 2012. p. 195-204.
- Wampler93 Wampler, J. L., Gunning, D. R., Wynkoop, C. E., Quill, L. L., & Moorman, L. L. (1993). Human Computer Interface Specifications (HCIS) for the Integrated Maintenance Information System (IMIS) (No. AL/HR-TP-1993-0035). ARMSTRONG LAB WRIGHT-PATTERSON AFB OH HUMAN RESOURCES DIRECTORATE.
- Ward08 Ward M., Gaynor D., Nugent T., Morrison R., HILAS Maintenance Solutions; challenges and potentials for the aircraft maintenance industry, 2008 IET Seminar on Aircraft Health Management for New Operational and Enterprise Solutions, 2008, 1-7.
- Wandt12 WANDT, Karina; TRETTON, P.; KARIM, Ramin. Usability aspects of eMaintenance solutions. Proceedings of eMaintenance, p. 12-14, 2012.
- Weld03 Weld, D. S., Anderson, C. R., Domingos, P., Etzioni, O., Gajos, K., & Lau, T. A. et al.: Automatically personalizing user interfaces. In International joint conference on artificial intelligence, 2003, 1613–1619.
- Westerfield12 WESTERFIELD, Giles. Intelligent augmented reality training for assembly and maintenance. 2012.
- Wickens08 Wickens, C. D. (2008). Situation awareness: Review of Mica Endsley's 1995 articles on situation awareness theory and measurement. Human Factors, 50(3), 397-403.
- Wooldridge09 Wooldridge, M., 2009. An introduction to multiagent systems. John Wiley & Sons.

- Wulf08 WULF, Volker; PIPEK, Volkmar; WON, Markus. Component-based tailorability: Enabling highly flexible software applications. *International Journal of Human-Computer Studies*, v. 66, n. 1, p. 1-22, 2008.
- Yang15 YANG, Tao. Effects of display position on guided repair and maintenance assisted by head-mounted display (HMD). 2015. Thesis (Georgia Institute of Technology)
- Yau02 YAU, Stephen S.; WANG, Yu; KARIM, Fariaz. Development of situation-aware application software for ubiquitous computing environments. In: *Computer Software and Applications Conference, 2002. COMPSAC 2002. Proceedings. 26th Annual International*. IEEE, 2002. p. 233-238.
- Yim10 YIM, Ho Bin; SEONG, Poong Hyun. Heuristic guidelines and experimental evaluation of effective augmented-reality based instructions for maintenance in nuclear power plants. *Nuclear Engineering and Design*, v. 240, n. 12, p. 4096-4102, 2010.
- Yim11 YIM, Ho Bin; KIM, In; SEONG, Poong Hyun. An Abstraction Hierarchy based mobile PC display design in NPP maintenance considering the level of expertise. *Nuclear Engineering and Design*, v. 241, n. 5, p. 1881-1888, 2011.
- Yim13 YIM, Ho Bin; KIM, Ar Ryum; SEONG, Poong Hyun. Development of a quantitative evaluation method for non-technical skills preparedness of operation teams in nuclear power plants to deal with emergency conditions. *Nuclear Engineering and Design*, v. 255, p. 212-225, 2013.
- Yim14 YIM, Ho Bin; LEE, Seung Min; SEONG, Poong Hyun. A development of a quantitative situation awareness measurement tool: Computational Representation of Situation Awareness with Graphical Expressions (CoRSAGE). *Annals of Nuclear Energy*, v. 65, p. 144-157, 2014.

Appendix A - GDTA FOR MAINTENANCE

High-level Goal: keep the factory/powerplant operational, with minimum downtime and maximum equipment availability, while also guarantying safety for everyone and for assets.

Main goal	Goal	subgoal	Decision/action	SAW Information requirement
1. Prepare for the maintenance work	1.1 Identify task risks and ways to mitigate them	1.1.1 Identify and propagate risks of the task to the team	1.1.1.1 Decision: What are the risks to be prevented?	<ul style="list-style-type: none"> 2- Risks to the person and equipment 2- Condition of the equipment <ul style="list-style-type: none"> 1- Sensor measurement 1- Variable boundary (of the sensor) 1- Risks measured by this sensor 2- Risks forecasted to the equipment (FMEA, HAZOP, fault tree) <ul style="list-style-type: none"> 1- When this risk occur 1- Likelihood 1- Severity 1- Consequence 1- Classification (Physical/Chemical/Ergonomical risks/etc) 2- Risks to the person <ul style="list-style-type: none"> PCMSO –RN-7 (Program of medical control of occupational health) 2- Risks of the environment <ul style="list-style-type: none"> 2- Risks forecasted of the place - RN-5 (LAIA or Gathering of Environmental Aspects and Impacts) <ul style="list-style-type: none"> 1- Likelihood 1- Severity 1- Consequence 1- Classification (Physical/Collective/Security/etc...) 2- Environment condition (insalubrity, others) <ul style="list-style-type: none"> 1- Visibility 1- Noise 1- Weather 1- physicochemical/physical security/collective safety

		<p>1.1.2 Reduce/remove risks</p>	<p>1.1.2.1 Decision: Which equipment should be isolated to execute the task?</p>	<p>Before (execution) 2- Equipment to be isolated 2- Connection between equipment 1- Layout diagrams 2- Energy sources 1- Electric 1- Hydraulic 1- Pneumatic 1- Physical position of equipment 1- Procedure to be executed and required intervention on the production line</p> <p>During (execution) 3- Impact of isolation the equipment 2- Equipment to be isolated 2- Tasks in isolated equipment 1- Scheduled tasks in isolated equipment 1- current tasks in isolated equipment</p>
			<p>1.1.2.2 Decision: Which equipment and tool should be used to prevent personal and environmental risks?</p>	<p>2- List of PPE and CPEs for a task Callout: “1.1.1.1 Decision: What are the risks to be prevented?” Callout: “1.1.2.1 Decision: Which equipment should be isolated to execute the task?” 2- PPE/CPEs options to manage risk 1- Location of PPEs 1- Condition of PPEs 1- Which PPE need testing</p>
	<p>1.2 Provision tools and materials (spares)</p>		<p>1.2.1 Decision: Is there enough material (assets) to execute the task?</p>	<p>2- Required materials 2- List of required materials 1- Target equipment 1- Target problem 1- List of materials 1- Amount required 2- Stock of materials 1- Amount of spare parts 1- Condition of the spare parts 2- Time required to gather the material 1- Material location 1- Current user location 1- Travel time</p>

			1.2.2 Decision: Which tool is required to execute the work?	2- Required tools 1- Target equipment 1- Procedure 1- Tools
2. Repair equipment	2.1 Make sure the equipment is safe to start tasks	2.1.1 Check the safety of the procedure and equipment	2.1.1.1 Decision: Is it safe to start operating the equipment?	3- Projection of PPEs and CPEs Callout: "1.1.2.2 Decision: Which equipment and tool should be used to prevent personal and environmental risks?" 2- State of each PPE/CPE 1- Equipped 2- Placement 1- Current place 1- Where it should be equipped/deployed 2- Need testing 1- Tested 2- Equipment isolation Callout: "1.1.2.1 Decision: Which equipment should be isolated to execute the task?" 2- List of isolated equipment 3- Impact of starting the task on people nearby 1- People nearby 1- Equipment isolation 2- Lockout status 3- Consequence of procedure on the equipment
			2.1.1.2 Decision: Where should the Collective Protective Equipment be deployed?	2- Ideal CPE distribution 1- Plant layout 1- Tasks place 1- Range of cover of the CPE
			2.1.1.3 Decision: What are the possible risks to be aware?	Risk by time (insalubrity) 3- Best time to take a rest 2- Time remaining to deadline 1- Time since the beginning of the task 1- Value of the pertinent variable (noise) 1- Maximum time of allowed work

				<ul style="list-style-type: none"> 1- Estimated time to leave the place 1- Estimated time of interruptions 1- Task total duration 1- Remaining task duration time 1- Shift total time 1- Remaining shift time 1- Procedure interruptibility
				<ul style="list-style-type: none"> Risk by action (stairs, ergonomics, dangerous equipment) <ul style="list-style-type: none"> 2- Protection mode <ul style="list-style-type: none"> 1- Risk classification 1- Source 1- PPE/CPE
				<ul style="list-style-type: none"> Risk by presence (height, confined space, oil - fumes) <ul style="list-style-type: none"> 1- Problem location 1- Protection equipment 3- Consequences of actions
				<ul style="list-style-type: none"> Risks (physical, chemical and biological)
		2.1.2	2.1.2.1	<ul style="list-style-type: none"> 3- Impact of automation on the target equipment <ul style="list-style-type: none"> 1- Equipment position 2- Automations in the target equipment <ul style="list-style-type: none"> 1- Type of automation <ul style="list-style-type: none"> 1- physical actions of the automation 1- virtual actions of the automation
	2.2	2.2.1	2.2.1.1	<ul style="list-style-type: none"> 2- Potential parts/pieces with problems (FMEA, maintenance planning and Work Instructions) <ul style="list-style-type: none"> 2- Visible age cues <ul style="list-style-type: none"> 1- Wear, leakages, etc 2- Factors for observations <ul style="list-style-type: none"> 1- sensor 1- thermal vision 1- noise produced by the equipment 1- Parts near their preventive maintenance 1- Problems reported by operators 1- Identification/probability/severity (severity matrix) 1- Manufacturer information
	Identify and diagnose equipment problems	Predictive maintenance: Make sure that the equipment will not present unexpected problems in the	Decision: What automations should be acknowledged during the task?	Decision: Which tests have to performed in the equipment?
		Check automations involving equipment		

		near future		<ul style="list-style-type: none"> 2- Time available for tests <ul style="list-style-type: none"> 1- Test duration 2- Time available to stop the equipment <ul style="list-style-type: none"> 1- Daily plans (schedule)
		2.2.2 identify problems	2.2.2.1 Decision: Which equipment should be inspected?	<ul style="list-style-type: none"> 3- Impact of not doing the inspection <ul style="list-style-type: none"> 2- Cost benefit <ul style="list-style-type: none"> 2- Inspection cost (time) <ul style="list-style-type: none"> 1- Value of time for the personnel involved in the activity 1- Estimated time of inspection 2- Cost of corrective maintenance of equipment <ul style="list-style-type: none"> 1- Value of time for the personnel involved in the activity 1- Time estimated repair 1- Equipment reliability 1- Fault tree analysis / hazop 1- Identification/probability/severity (Severity Matrix) 2- Expected condition of the equipment <ul style="list-style-type: none"> 1- Last inspection performed on the equipment 1- Abnormalities list by the operator 1- Current sensor measurement 1- Expected sensor measurement 2- Current Age of equipment <ul style="list-style-type: none"> 1- Expected wear 1- Lifetime 1- Signs / cues of age (real wear) 2- Regulation <ul style="list-style-type: none"> 1- Law (regulatory standards) 1- Company policies
			2.2.2.2 Decision: How to inspect the equipment?	<ul style="list-style-type: none"> 2- Expected condition of the equipment <ul style="list-style-type: none"> 1- Last inspection performed on the equipment 1- Abnormalities list by the operator 1- Current sensor measurement 1- Expected sensor measurement 2- Current Age of equipment <ul style="list-style-type: none"> 1- Expected wear 1- Lifetime 1- Signs / cues of age (real wear)

		2.2.3 Diagnose the problem	2.2.3.1 Decision: What is the equipment defect? (diagnosis of any failure/defect of the equipment)	<ul style="list-style-type: none"> 2- Problem identification <ul style="list-style-type: none"> 1- Operator reports (abnormalities) 2- Analysis of every part (piece) of the equipment 2- Analysis of operational diagrams <ul style="list-style-type: none"> 1-Electric diagram 1-Pnematic diagram 1- Instruction manual 2- Equipment history <ul style="list-style-type: none"> 1- Maintenance history 1- Last test history 1- Common problems of this equipment 2- Possible source of problems <ul style="list-style-type: none"> 1- Fault tree analysis 1- Sensor history information
	2.3 Understand the problem and present a solution	2.3.1 Classify the type of maintenance	2.3.1.1 Decision: Is it necessary to fix the equipment now (classify the maintenance)?	<ul style="list-style-type: none"> 3- Impact of stopping the equipment in the production line 2- Cost of the repair 2- Time required to repair the problem 2- Length of time item can be deferred without repair 2- Risk of operating the equipment in current state Callout: "1.2.1 Decision: Is there enough material (assets) to execute the task?" 2- Work planning for today, this week and this month 2- Cost and risk of aggregated equipment Callout: "2.2.2.1 Decision: Which equipment should be inspected?" Callout: "2.2.3.1 Decision: What is the equipment defect?" <p>Opportunistic maintenance:</p> <ul style="list-style-type: none"> 1- Maintenance capacity of the destined factory
		2.3.2 Study the problem and propose a solution	2.3.2.1 Decision: How to solve the problem?	<ul style="list-style-type: none"> 3- Impact of potential approaches on time 3- Impact of potential approaches on operational safety 3- Impact of potential approaches on other tasks/jobs <ul style="list-style-type: none"> 2- Possible methods 2- Possible sources of problem 2- Maintenance/failure history of item Callout: "1.2.1 Decision: Is there enough material (assets) to execute the task?"
			2.3.2.2 Decision: How to dis/assemble the	

			equipment?	
			2.3.2.3 Decision: How should the equipment be repaired?	Callout: "2.2.3.1 Decision: What is the equipment defect?" 1- Procedure for fixing the problem 1- Equipment CAx
	2.4 Respond to abnormalities		2.4.1 Decision: What are my teammates doing that I need to be aware of?	3- Impact of teammate activity on my work 2- Current situation/status of teammate 2- The procedure 1- Target Equipment 1- Concurrency of the procedure 1- Risk of the procedure 1- Location 1- Video Feed
		2.4.2 Decision: Is there an incident principle?	Physicochemical: Incident 3- Projected risk of the environment 2- Current condition 1- Sensor measurement 2- Safety Boundary 1-Value 1-Direction (above/below expected) 1- Trend of the sensor (increase/decrease)	
			2- Current environment condition 1- Visibility 1- Noise 1- Weather 2- physicochemical 1- Current sensor measurement 1- Expected sensor measurement 1- Variable trend 2- Safety limit 1-Value 1-Direction (above/below expected) 1- Physical security 1- Collective safety	

3. Prepare equipment for production line	3.1 Make sure the equipment works after the repair		3.1.1 Decision: Can the equipment be released for normal execution?	2- Consequence of reconnecting the equipment to the production line
			3.1.2 Decision: Is it possible to remove isolation and reintegrate the equipment?	3- Impact of removing isolation from the equipment 1- Isolated equipment 1- Special care for turning on the equipment 2- Production line and impact on other equipment 2- Field personnel that may be impacted 1- Work schedule on the impacted equipment 1- Work order in progress 1- Locations under work
	3.2 Ensure that staff and others are aware that the work is finished		3.2.1 Decision: Who needs to be informed of the end of a task?	1- Operator responsible for the equipment 1- Team members
	3.3 List problems / difficulties encountered at work		3.3.1 Decision: Does the workplace needs cleaning after the task?	3- Possible consequences and risks of keeping the generated waste 1- Waste and liquid spread in the workplace 1- Cleaning personnel available 1- Disposal waste site that is safe and appropriate for the environment according to regulations (disposal for plastic, metal, paper, toxic waste, etc.)
4.			4.1.1 When is the time for a shift handover?	2- Best time for shift handover 1- Total shift time 1- Current shift time

Appendix B - SPECIALIZED DECISION COMPONENTS

B.1 Procedure Inspection Specialized Decision Component

There are several ways to model procedures in the literature (HTA, IETM), some specific for maintenance [Stock05]. Any procedure modeling technique could be adapted to create Procedure Decision Components, but for the study case a modeling technique based on classification was chosen. The inspection procedure was studied to have a SpecDC defined to speed up agents modeling.

The decision modeled here is “How to inspect the equipment?”, and the information requirement is:

- 2- Expected condition of the equipment
 - 1- Last inspection performed on the equipment
 - 1- Abnormalities list by the operator
 - 1- Current sensor measurement
 - 1- Expected sensor measurement

What maybe the hardest part for designers is the organization of the procedure in groups. A task usually consists of several small procedures, but they have to be grouped so that the UI is only showing some of them at a time. For the study case, we defined a quantity of four procedures per round, so that the 28 procedures were organized in 7 steps.

Concerning inspections of equipment, four variables relative to the equipment inspected can be used to define an inspection procedure. These variables, when delegated to a procedure, will assist in a fast creation of their UIs.

The first variable is about sensing possibilities and each equipment can be either: S) has a sensor; I) has an indirect sensor; O) organizational (has no sensor).

The second variable is related to the nature of the problem to be inspected and values can be: D) defect; A) abnormality. The third variable is about the quantity of equipment to be inspected and values can be: U) it is a unique equipment; G) it is group or collection of equipment; E) it is not an equipment, but something around the environment. Finally the fourth variable is related to the cue of the problem and values can be: V) visual; A) auditory.

Using these four variables, designers can quickly model a Decision Component for a procedure agent for different inspections.

As an example, pretend to model the inspection of a Slip Ring, to check if it is overheating. Since overheating is about temperature, something that can be measured by sensors, the first variable is S (equipment has a sensor). The second variable is A (abnormality), because users need to check if there is something unusual with the equipment, and not if it is broken (if this equipment was broken the plant would have stopped). The third variable is U (unique), because there is only one slip ring, and not a set of slip rings. The fourth variable is V (visual), because the inspecting is made through looking at the equipment and identifying abnormal coloration (red), which indicates the overheating.

With some inspections already modeled, the common or similar inspections were grouped and representation components were defined. Figure B-1 shows the three representation components created for this SpecDC:

- Equipments with sensor: since a sensor is the center of the inspection, the current sensor value is the center of the representation component. Example: slip ring;
- Equipment with no sensor (or indirect sensor): for these inspections, the UI focus on showing a functional and a non-functional equipment. Example: lamp;
- Equipment with disruptions: for these cases, the UI show a symbol of the possible disruption in the equipment. Example: wires (broken).

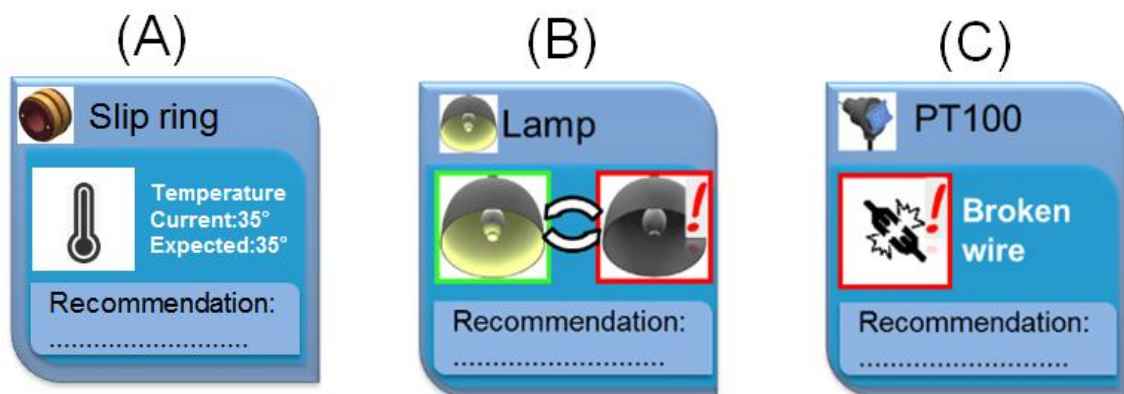


Figure B-1: Three representation components created for the equipment inspection SpecDC.

The final step is to link the procedure to another agent. These inspections procedures are normally to inspect an equipment or environment, and this linking is what enables other agents to find out what is the current equipment or environment under inspection.

SpecDCs to define other maintenance activities, such as testing, diagnosing, repairing and assembling, could also be created. Some of these procedures, like diagnostic and repair, are already thoroughly described in several papers, considering modeling of UI obtaining data from the equipment (CAD data) [Fiorentino16, Espindola13].

B.2 Team Inspection Specialized Decision Component

Another SpecDC created was for Team agents and communication and decisions of users regarding this situation aspect during equipment inspection.

The decision modeled here is “What are my teammates doing that I need to be aware of?”, and the information requirement are:

- 3- Impact of teammate activity on my work
 - 2- Current situation/status of teammate
 - 1- The procedure
 - 1- Target Equipment
 - 1- Concurrency of the procedure
 - 1- Risk of the procedure
 - 1- Location
 - 1- Video Feed

For this SpecDC, three elements every Team agent needs to define are: overall procedure synchronization among teammates, concurrency locks and risky procedures.

Synchronization and coordination among teammates will determine the rate of updates and level of details that users are informed about each team member. In a tightly synchronized team, every teammate knows the exact step of the procedures the others are, and they need to coordinate their actions using this knowledge. On the other hand, team coordination can also be loose, whereupon teammates only know vague details of the others.

Concurrency locks are blocking procedures that require only one user involved with an equipment. Because researchers have noted that locks are less important when there is awareness of what objects people are working upon [Gutwin02], and that social protocols can be used to coordinate when awareness is given [Greenberg94], these locks will be used for loose coordination, and not tight.

In a tight synchronized scenario people will already know the objects being worked by their teammates, as specified. But in a loose synchronized scenario, in some occasions, teammates will decide to work on an object when it is occupied. For this scenario, the lock will be useful to immediately warn the user that the object is indeed occupied. That is why locks are only showed in loose synchronized teams.

Risky procedures are procedures that may offer risks to others.

Concurrency and risky both need to be modeled per procedure, and Table B-1 shows an illustrative example of the table.

Table B-1: Concurrent locks and risky procedures modeling.

	P1	P2	P3
Concurrency	parallel	lock	lock
Risks to others	none	Dangerous equipment being tested	none

Using this SpecDC, a team network will be simulated by the team agents, passing information from one another that are important for users inside and sometimes outside the team.

SpecDCs can also use Messaging mechanism to communicate/cooperate with other agents. For instance, for the Team inspection SpecDC, P2 and P3 were defined as blocking procedures (Table B-1). Therefore, to warn other users, a Messaging schematic was defined to activate whenever a person in the team starts a blocking procedure.

The pseudo-algorithm used for this is: If *current Procedure concurrency* is of the type *lock* and another *Person* will start a *Procedure* in the same *equipment*, then activate trigger.

Two Messaging schematic were defined for these case, using the mechanisms PassMessageToTeammate and AssumeResponsibility. These set of Messaging schematics means that whenever procedure P2 or P3 are started by the user, all his/her teammates receive a notification that the equipment inspected in this procedures cannot be used.

Another Messaging schematic was defined to happen when a teammate feels sick, to warn everyone. Even other coworkers that are not in the referred team, but are close to the task location, are warned by these Messages. This creates a theoretical network between teams, where important information can be passed among them if they are connected.

Figure B-2 demonstrates a simple team network case, in which team A is connected to B and C due to proximity. Team A and C have a loose synchronization, while team B is tightly synchronized. Team C is also connected to D, which is just a passerby (considered as an “independent” team), but any risk on team C tasks are passed on to D.

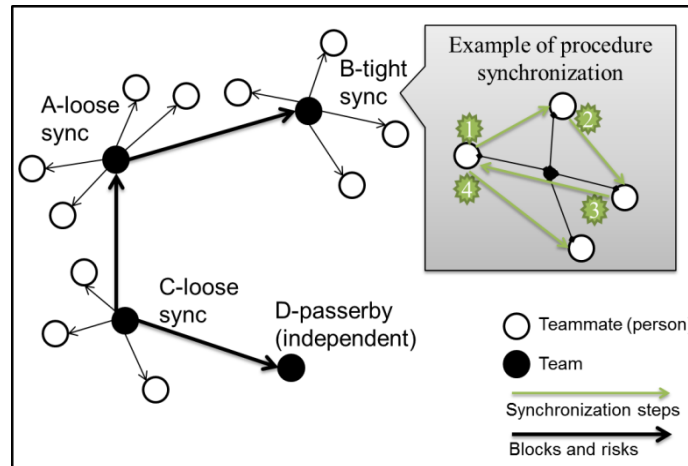


Figure B-2: Example of a team network formed by proximity to warn others teams of risks and blocking procedures.

Figure B-2 also shows an example of procedure inside team B, the only team that needs tight synchronization. The green arrows show the sequence of procedure that needs to be executed and by which person. An example of tight synchronization procedure could be the disassembly of a large piece of equipment.

Since this SpecDC is based on procedures (Table B-1), a Team agent would have many copies of this Decision Component in it, one for each task (with many procedures). For instance, an enterprise with 5 teams, in which each team works on 10 inspections tasks, would have 5 Team agents, each with 10 Team Inspection SpecDC.

Additionally, there are other problems considering Team Awareness that are not covered by this SpecDC, like long distance cooperation and shifts handovers.

B.3 Personal Protective Equipment Specialized Decision Component

This SpecDC was created to assist the following decisions, with their respective information requirements:

- What are the risks to be prevented?
 - 2- Risks of the environment
 - 2- Risks forecasted of the place - RN-5 (LAIA or Gathering of Environmental Aspects and Impacts)
 - 1- Likelihood
 - 1- Severity
 - 1- Consequence
 - 1- Classification (Physical/Collective/Security/etc...)
 - 2- Environment condition (insalubrity, others)

- 1- Visibility
 - 1- Noise
 - 1- Weather
 - 1- physicochemical/physical security/collective safety
- Which equipment and tool should be used to prevent personal and environmental risks?
 - 2- List of PPE for a task in an equipment
 - Callout: “1.1.1.1 Decision: What are the risks to be prevented?”
 - 2- PPE options to manage risk
 - 1- Location of PPEs
 - 1- Which PPE need testing
- Is it safe to start operating the equipment?
 - 3- Projection of PPEs and CPEs
 - Callout: “1.1.2.2 Decision: Which equipment and tool should be used to prevent personal and environmental risks?”
 - 2- State of each PPE/CPE
 - 1- Equipped
 - 2- Placement
 - 1- Current place
 - 1- Where it should be equipped/deployed
 - 2- Need testing
 - 1- Tested

The process of modeling this SpecDC is based on choosing a risk and associating a protective gear to deal with it.

Figure B-3 (A) shows the presentation defined for this Decision Component. Designers fill a structure with risks and protective gears (linking the Equipment agent to this Decision Component).

Then, during run-time, this component is constantly checking the agents, and after each equipment is gathered, their icon becomes semi-transparent (0.4 alpha), as shown in Figure B-3 (B). Finally, after gathering an equipment that need testing, a shaking hand is showed to users until they test the equipment (and only then their icon becomes transparent).

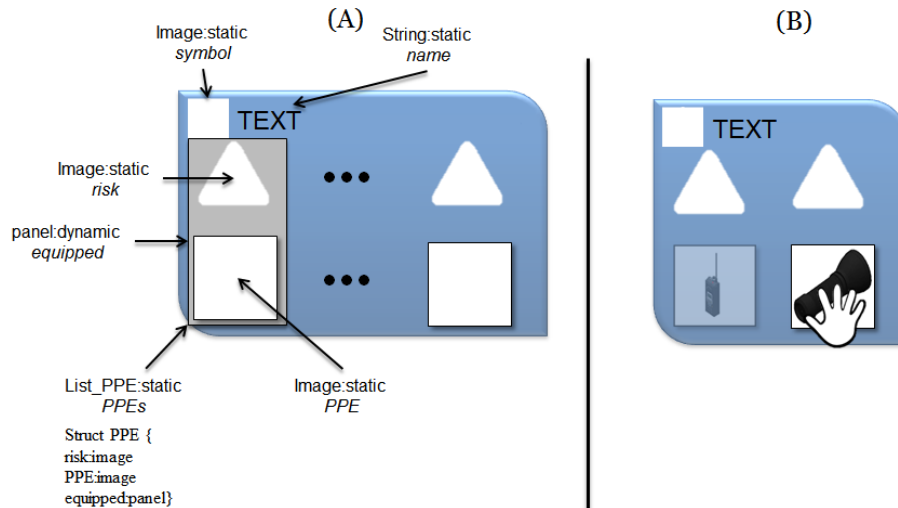


Figure B-3: UI of the Personal Protective Equipment Specialized Decision Component.

B.4 Collective Protective Equipment Specialized Decision Component

This SpecDC was created for the decision: “Where should the Collective Protective Equipment be deployed”

- 2- Ideal CPE distribution
 - 1- Plant layout
 - 1- Tasks place
 - 1- Range of cover of the CPE

For this components, designers choose a CPE and define where they should be deployed, using a blue print (or up view picture) of the task place. Figure B-4 shows the base of the UI (A) and an example (B) using cones as CPE (with metadata).

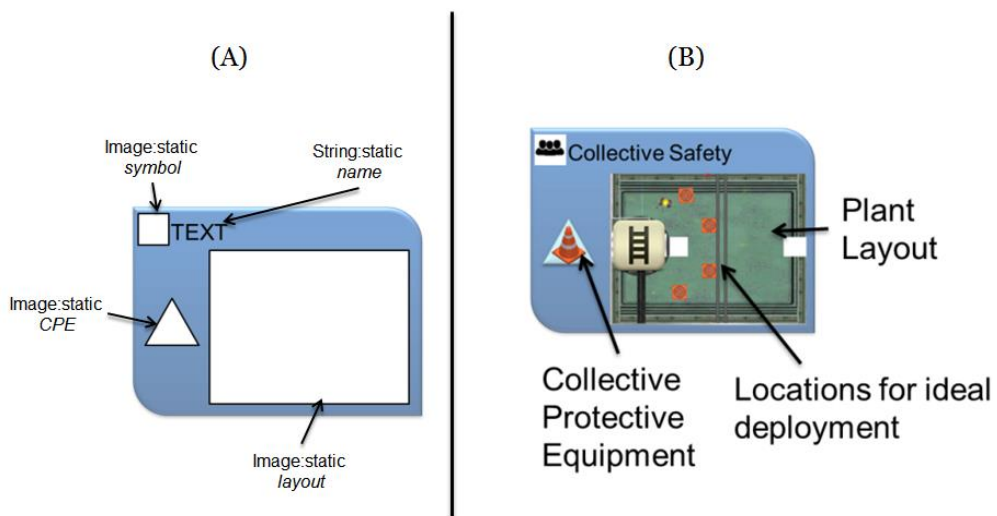


Figure B-4: UI of the Collective Protective Equipment Specialized Decision Component.

B.5 Inspection Maintenance Classification Specialized Decision Component

This SpecDC was created for the decision: “Is it necessary to fix the equipment (categorize the type of maintenance)”

- 3- Impact of stopping the equipment in the production line
 - 2- Cost of the repair
 - 2- Time required to repair the problem
 - 2- Risk of operating the equipment in current state

This Decision Component uses a fault tree analysis to structure each possible equipment problem and their consequences, assigning cost, risk and impact to each problem (and possible combinations and consequences).

Cost varies from 1 to 10, while risk from 1 to 5 (traditional measure of risk from Failure Mode and Effects Analysis - FMEA). Impact is defined as the consequence in the production line of stopping this equipment for reparation. Designers define the cost, risk and impact of each problem, and can associate two (or more) problems as having greater cost/risk when combined.

Figure B-5 demonstrates the association between the information requirement for this decision and its UI using an imaginary example:

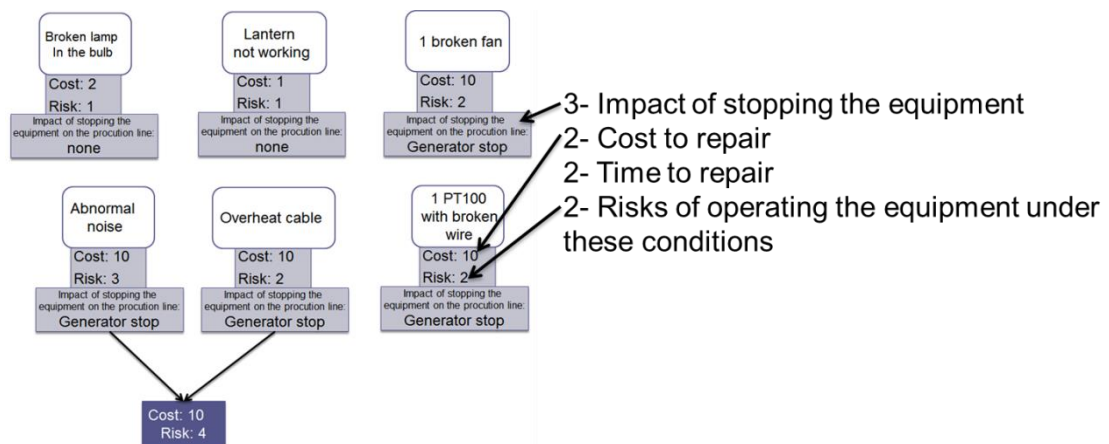


Figure B-5: Example of UI for the Inspection Maintenance Classification Specialized Decision Component.

Notice that, for this decision, a deliberative process will be used by the user, instead of a more rapid situation recognition pattern matching process, characterized in the Recognition-Primed decision making theory [Klein93], thus the huge difference in UI style. A diagnosis procedure would also use a more deliberative process and have a UI that could better support it.

Appendix C - LIST OF ERRORS COMMITTED IN THE SIMULATION

Table C-1 lists all errors committed and their corresponding lack of awareness. Bold/underlined errors indicate errors regarding an unsafe behavior.

Table C-1: List of possible errors in the simulation of inspecting a hydropower plant generator and their corresponding Awareness.

General	Proc	Equip	Team	Ent	Env	Experts Guide- UI	Experts SAW- UI	Non- experts SAW- UI
Interrupted the task without a cause (with a 1h or 1 day rest)				x	x	0	0	0
<u>Did not attach the belt before going up stairs</u>	x				x	5	1	0
Regulations								
<u>Did not evacuated in case of imminent accident (continued working without noticing the problem)</u>				x	x	18	0	0
<u>Did not rest due to WBG T</u>				x	x	10	1	0
<u>Did not rest due to noise</u>				x	x	9	1	1
<u>Did not assist coworker during accident</u>			x	x		10	8	0
Did not coordinate with the team to execute another task when the equipment was being used			x			9	1	1
Procedure 1								
<u>Did not take a PPE</u>		x		x		3	0	0

<u>Got a wrong PPE</u>		x		x		2	0	1
<u>Did not test the lantern</u>		x				2	0	0
<u>Did not test the backup lantern</u>		x				2	2	0
Did not report (or incorrectly reported) broken lantern	x	x				2	0	0
Procedure 2								
<u>Did not use the cones for collective protection</u>		x		x		0	1	2
Did not report (or incorrectly reported) cleaning problems in the generator cover	x	x				0	0	0
Did not report (or incorrectly reported) broken lamp in the generator cover	x	x				0	0	1
Procedure 3								
<u>Did not attach belt before going down stairs</u>	x				x	1	2	0
Procedure 4								
Did not report (or incorrectly reported) cleaning problems in the bulb	x	x				0	0	0
Did not report (or incorrectly reported) broken lamp in the bulb	x	x				0	0	0
Did not report (or incorrectly reported) abnormal noises	x	x				0	0	0
Did not report (or incorrectly reported) broken fan	x	x				0	0	0
Procedure 5								
Did not report (or incorrectly reported) abnormal vibrations in	x	x				0	0	0

the cylinder head								
Did not report (or incorrectly reported) oil leakage in the in the cylinder head	x	x				0	0	0
Did not report (or incorrectly reported) broken wire in a PT100	x	x				0	0	1
Did not report (or incorrectly reported) PT100 overheating	x	x				1	3	1
Procedure 6								
Did not report (or incorrectly reported) overheat in feeding cables A and B	x	x				0	0	0
Did not report (or incorrectly reported) sparking in the slip ring	x	x				0	0	0
Did not report (or incorrectly reported) overheat in the slip ring	x	x				1	3	0
Did not report (or incorrectly reported) abnormal vibrations in the slip ring	x	x				5	4	0
Procedure 7								
Did not report (or incorrectly reported) wear in one of the brushes of a brush holder	x	x				0	0	0
Did not report (or incorrectly reported) misalignments in one of the brushes of a brush holder	x	x				0	0	0
Did not report (or incorrectly reported) broken wire in a tail	x	x				0	1	0
Did not report (or incorrectly reported) overheat in a tail	x	x				0	0	0

Appendix D - QUESTIONNAIRE

1. Participant information

- a. Name: _____
- b. Job: _____
- c. Years with experience with industrial maintenance: _____

2. Experience with computer and games

a. What is your level of knowledge/ability of the computer?

- 1: Very poor 4: Good
- 2: Poor 5: Very good
- 3: Average

b. What is your level of knowledge/ability of games and virtual worlds?

- 1: Very poor 4: Good
- 2: Poor 5: Very good
- 3: Average

3. Experience with the activity of inspecting the generator

a. About this activity of inspecting the generator:

- 1: I do not know it
- 2: I know the basics of it
- 3: I know it very well

b. About your experience of inspecting the generator::

- 1: Never done it
- 2: Never done it myself, but watched it
- 3: I have done it many times – 1 to 5
- 4: I have done it plenty of times – 5 more

c. Do you know RN-5(CIPA)?

- 1: No 2: A little 3: Yes

d. Did you ever participated of a CIPA of a hydro powerplant, or other company with high risk maintenance?

1: Yes

2: No

e. Did you ever made a training about job safety and regulator norms that effect your profession?

1: Yes

2: No

3: Maybe (explain): _____

4. Simulation validation

a. In your opinion, was the fidelity of the procedure in the simulation high?

1: Strongly agree

4: Disagree

2: Agree

5: Strongly disagree

3: Do not agree or disagree

b. In your opinion, was visual and sound fidelity of the defects in the simulation high?

1: Strongly agree

4: Disagree

2: Agree

5: Strongly disagree

3: Do not agree or disagree

5. Comparison of interfaces: Guide-UI and SAW-UI

a. Is SAW-UI is easy to understand and use.

1: Strongly agree

4: Disagree

2: Agree

5: Strongly disagree

3: Do not agree or disagree

b. Using SAW-UI, I can remember where every type of information is located in the screen.

1: Strongly agree

4: Disagree

2: Agree

5: Strongly disagree

3: Do not agree or disagree

c. SAW-UI would make my jog easier then Guide-UI, if both were adjusted to daily tasks.

1: Strongly agree

4: Disagree

2: Agree

5: Strongly disagree

3: Do not agree or disagree

d. SAW-UI would help lessen risks (to you, to others and to equipment) better than Guide-UI.

1: Strongly agree

4: Disagree

2: Agree

5: Strongly disagree

3: Do not agree or disagree

e. SAW-UI showed an ideal quantity of information in the screen.

1: Strongly agree

4: Disagree

2: Agree

5: Strongly disagree

3: Do not agree or disagree

Number 6 was only asked to non-experts

6. About the division of the interface in 5 entities:

E. Procedure: the interface helped me understand step by step of the procedure.

1: Strongly agree

4: Disagree

2: Agree

5: Strongly disagree

3: Do not agree or disagree

F. Equipment: the interface helped me understand the condition of the equipment to be inspected.

1: Strongly agree

4: Disagree

2: Agree

5: Strongly disagree

3: Do not agree or disagree

G. Team: the interface helped me understand the condition of each team member.

1: Strongly agree

4: Disagree

2: Agree

5: Strongly disagree

3: Do not agree or disagree

H. Environment: the interface helped me understand the condition of the environment.

1: Strongly agree

4: Disagree

2: Agree

5: Strongly disagree

3: Do not agree or disagree

I. Enterprise: the interface helped me understand rules and regulator norms of work required by the company (such as WBGT, noise and evacuation in accidents).

Guide-UI:**SAW-UI:**

j. Quantity of information: In average, during the simulation information came with enough volume to help you understand the situation (high) or not (low).

Guide-UI:**SAW-UI:**

k. Quality of information: In average, during the simulation, information had high and trustworthy quality (high) or not (low).

Guide-UI:**SAW-UI:**

l. Familiarity with the situation: In average, during the simulation you felt lost (low) or you knew what you were doing (low).

Guide-UI:**SAW-UI:**

m. Understanding: In average, during the simulation the interface helped you understand your situation (high) or not (low).

Guide-UI:**SAW-UI:**

n. Situation awareness: how high was your situation awareness? In average, during the simulation it was possible to comprehend what was going on and even anticipate events before they happened (high) or not (low).

Guide-UI:**SAW-UI:**

Calculating SART-score: The overall SART score is calculated using the general formula:

$$\text{SART-score (SAW)} = \text{U} - (\text{D} - \text{S})$$

where: U = average understanding; D = average demand; S = average supply. The complete formula for SART score, considering questions as Q1 to Q14, and SAW-question (the calculated value based on Q14), is:

$$U = (Q10 + Q11 + Q12 + Q13) / 4$$

$$D = (Q1 + Q2 + Q3 + Q4) / 4$$

$$S = (Q5 + Q6 + Q7 + Q8 + Q9) / 5$$

$$\text{SAW-question} = -10 + 30 * (Q14/10)$$

$$\text{SART-pure} = U - (D - S)$$

$$\text{SART-final} = (13 * \text{SART-pure} + \text{SAW-question}) / 14$$

Complete formula extended: $(13 * (((Q10 + Q11 + Q12 + Q13) / 4) - (((Q1 + Q2 + Q3 + Q4) / 4) - ((Q5 + Q6 + Q7 + Q8 + Q9) / 5))) + (-10 + 30 * (Q14/10))) / 14$

The SAW-question value is based on the fact that SART results can vary from -10 to +20. The SART-pure value is equivalent to 13D-SART, the version of SART that does not have the final question (Q14) that provides the SAW-question value. The SART-final value was calculated based on the fact that the SAW value (calculated based on Q14) has the same weight of all other questions, thus a weight of 1/14.

Appendix E - PUBLICATIONS

E.1 Main

Oliveira, Allan; Araujo, Regina. A human centered perspective of E-maintenance. In WRVA'2012 – Workshop de Realidade Virtual e Aumentada, 2012.

Oliveira, A.; Araujo, R.; Jardine, A. A Human Centered View on E-Maintenance. Chemical Engineering Transactions, v. 33, p. 385-390, 2013.

Oliveira, A., Araujo, R., & Jardine, A. (2014, June). Human-Centered Interfaces for Situation Awareness in Maintenance. In International Conference on Human Interface and the Management of Information (pp. 193-204). Springer International Publishing.

Oliveira, A. C. M., Van Volkenburg, C. A., Jardine, A. K., & de Araujo, R. B. (2015, January). The electronic maintenance situational awareness interface. In 2015 Annual Reliability and Maintainability Symposium (RAMS) (pp. 1-6). IEEE.

E.2 Others

Araujo, R.B.; Rocha, R.V.; Oliveira, A.C.M.; Botega, L. WINDIS Lab The cyber world that surrounds us. SBC Journal on 3D Interactive Systems (JIS 2011), v. 2, p. 102-105, 2011.

Oliveira, Allan; Duarte, Fernando Vieira. NFA- $\delta\epsilon$: the Non-Deterministic Finite Automaton with Random ϵ -moves, Environment Awareness and Memory for Serious Games Modeling. In WRVA'2014 – Workshop de Realidade Virtual e Aumentada, 2014.

Oliveira, A., Caetano, N., Botega, L. C., & de Araújo, R. B. (2015). A Head-up Display with Augmented Reality and Gamification for an E-Maintenance System: Using Interfaces and Gamification to Motivate Workers in Procedural Tasks. In Human Interface and the Management of Information. Information and Knowledge in Context (pp. 499-510). Springer International Publishing.