Earth, Life & Social Sciences
Kampweg 5
3769 DE Soesterberg
P.O. Box 23
3769 ZG Soesterberg
The Netherlands

www.tno.nl

T +31 88 866 15 00
F +31 34 635 39 77

TNO report

TNO 2015 R11582

Design Patterns for Transparency and Joint Control of Highly Autonomous Adaptive Systems

Date	16 december 2015
Author(s)	Dr. G.M. te Brake Dr. J. van Diggelen
Copy no	
No. of copies	
Number of pages	29 (incl. appendices)
Number of appendices	
Sponsor	TNO
Project name	ERP HE Adaptive Automation
Project number	060.14382

Contents

1 Introduction

Human operators in control room settings face increasingly more automation in their work environments. Not long ago, operators steered and controlled the systems, understanding them in their nitty gritty details. Today, operators have turned into information managers supervising complete plants or multiple complex systems that to a large extent operate autonomously. This transition is taking place in many different domains, ranging from traffic management, maritime & offshore tasks, utility network management or naval operations. The role of supervisor of automated systems and intervener in situations where automation is suboptimal or fails, requires different competences, skills, and support systems to keep track of the enormous amount of data available due to the increase in sensors and models.

Figure 1 shows the generic model of automation used in this study. An environment (e.g. plant, ship, road infrastructure, UAV…) is controlled using a set of sensors and actuators. Control software is installed and manages this process. In regular circumstances this works fully autonomously. However, situations will occur that were not foreseen in the design phase of the software, sensors or actuators will break or malfunction, or the control software contains bugs. In these cases human intervention is required to prevent the system from breaking down.

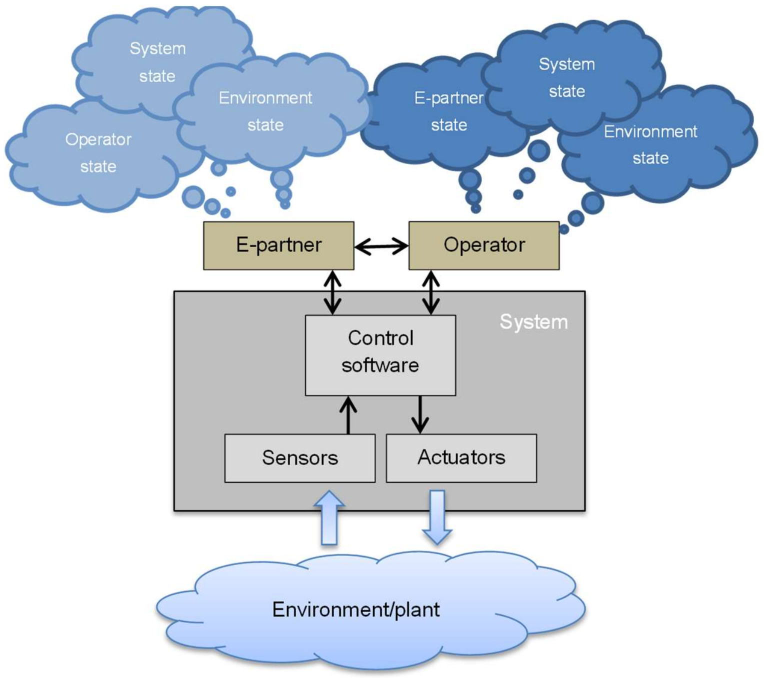


Figure 1: The model depicting the components of our generic human automation control system

An additional module of intelligent software, called an e-partner, could support the operator with this task. The e-partner can monitor the control software, the environment and the human operator. Based on internal models and the specifics of the situation, it can inform or advise the operator, tell the operator what to do, or could even adjust the control software itself. Although these developments started in the eighties with expert systems that advised on specific topics, the previous generation of support systems was not aware of the operator state and could not function as a colleague of the operator with varying (adaptive) work agreements. Joint control between human operator and intelligent e-partners in highly autonomous settings requires other types of support and a clear design of locus of control and transparency. Notice that this model consists of one operator and one system. This environment will be extended to mimic more complex operational environments in which teamwork is essential and multiple processes or environments are under control.

Part of the Early Research Program (ERP) Human Enhancement is a project that focusses on Adaptive Automation (AA). The ambition of this project is to develop a transparent (human-in-the-loop) adaptive automation platform, based on a computational human model to assess current and predicted human task load. The environments we focus on contain highly autonomous systems and often multiple processes under control (PuCs). Human supervision required is minimal, and great cost reductions can be made if the operator can conduct other tasks during quiet and predictable time periods. Hence, the aim of our adaptive automation concept is to enable operators to conduct other activities during stable periods of low risk. Our ambition is to lower the operational costs and increase efficiency, and to design a more interesting task set for the operator. To keep minimal situation awareness while conducting these other tasks, the operator receives status updates from the e-partner, and when required can return to the desk and takes over active control. To get the operator back in the loop as quickly as possible, the e-partner supports the quick development of situation and option awareness. Collaboration and locus of control is highly adaptive to the state of the system and the (task) environment, but also to the state (and location) of the operator.

To design this adaptive automation concept we distinguish three research topics:
1. Shared supervisory control: How to design adaptivity in human-e-partner collaboration for supervisory control tasks?
2. Minimal situation awareness (SA) levels: What is the minimal level of SA that the operator must have at all times to remain in-the-loop sufficiently and maintain safety, and which type of pro-active support should the e-partner provide to accomplish this in the most efficient / least intrusive way?
3. Just-in-time-awareness: What is the best way for the e-partner to support the operator to resume control, and how long does this take?

All three topics are related to each other, especially the latter two. When the continuously maintained minimum level of SA is kept at a higher level, the operator will be able take over control faster. However, a higher level of SA will imply more interruptions. Hence, a trade-off between the minimal level of SA and the availability for other tasks may have to be made. Or maybe in specific contexts the answer of research question 2 will determine the required minimal level of SA.

Chapter 2 will introduce the most relevant background knowledge on supervisory control. In a previous study (Van der Kleij et al., 2015), an extensive search of literature has been conducted. In this report, for ease of reading, the previously described theories and concepts used in the following chapters will be introduced shortly. In Chapter 3 an ontology is presented that contains the most important terms in our research, and will simplify reuse of concepts in other projects. Further, a generic Concept Development and Experimentation (CD&E) approach to develop supervising and intervening concepts for intelligent and effective human-system collaboration is presented. By embedding design patterns in a proven system engineering approach, a method is created to build human-automation concepts for the stated research questions. In Chapter 5 initial activities and concepts will be presented for a dynamic positioning use case, with a focus on research question number 2. A platform for demonstration and experimentation for this use case is described. The report ends with conclusions and implications for further work.

2 Background knowledge

2.1 Supervisory control and adaptivity

Our generic operational environment is built upon the work by Sheridan (2011, 2012) on supervisory control (Figure 2). In his representation, the e-partner is called the Pre-programmed Automatic Parameter Changer (PAPC), which can adjust the control logic of the software, but also the control or display interface to the human supervisor. The rules that define the allocation authority of the PAPC determine which parameters the PAPC may change. Our e-partner can be seen as an evolved version of the PAPC that also takes the operator and environment into account.

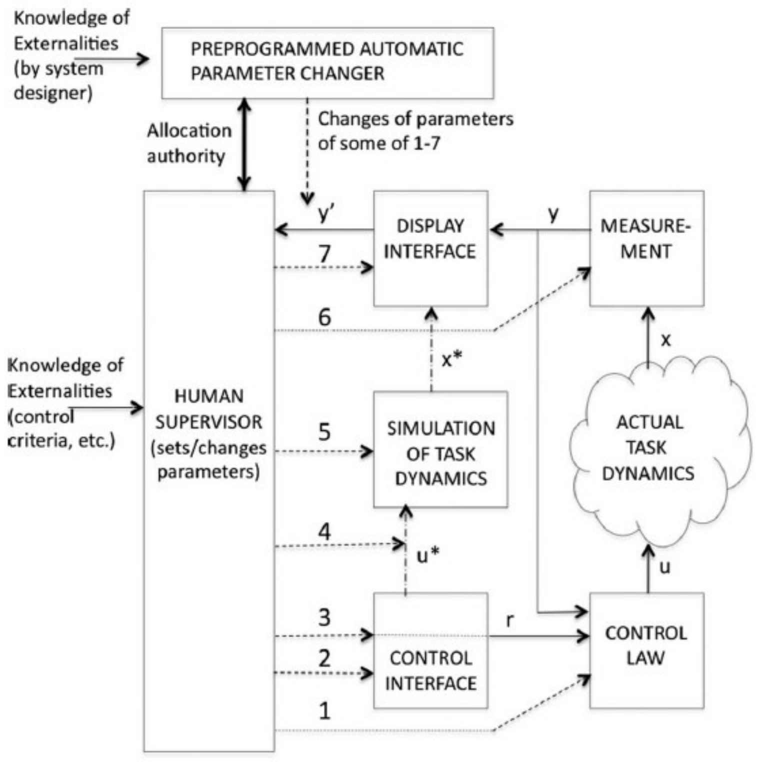

Knowledge of Externalities (by system designer)
PREPROGRAMMED AUTOMATIC PARAMETER CHANGER
Allocation authority
Changes of parameters of some of 1-7
y'
y
DISPLAY INTERFACE
MEASURE-MENT
7
6
x*
x
Knowledge of Externalities (control criteria, etc.)
HUMAN SUPERVISOR (sets/changes parameters)
5
SIMULATION OF TASK DYNAMICS
ACTUAL TASK DYNAMICS
4
u*
u
3
r
2
CONTROL INTERFACE
CONTROL LAW
1

Figure 2: Sheridan's model of supervisory control

A key concept in our approach is adaptive automation. Adaptation in systems can appear in different forms. We will adopt the taxonomy developed by Feigh, Dorneich and Hayes (2012) presented in Figure 3. This taxonomy distinguishes four different categories:

1. Modification of Function Allocation
2. Modification of Task Scheduling
3. Modification of Interaction
4. Modification of Content

Function allocation, dynamically shifting the locus of control of specific functions between a human operator and an intelligent system, is what is typically meant in studies of adaptive automation and relates to Sheridan's *Allocation authority*.

However, in this study we will adopt a wider scope and include the other three types of modification of adaptivity as well. Modification of interaction and modification of content are common in adaptive or intelligent user interface concepts, and change the *control* or *display interface* to the human operator. Modification of task scheduling means altering the process that is being controlled, i.e. the *control law* in Sheridan's model.

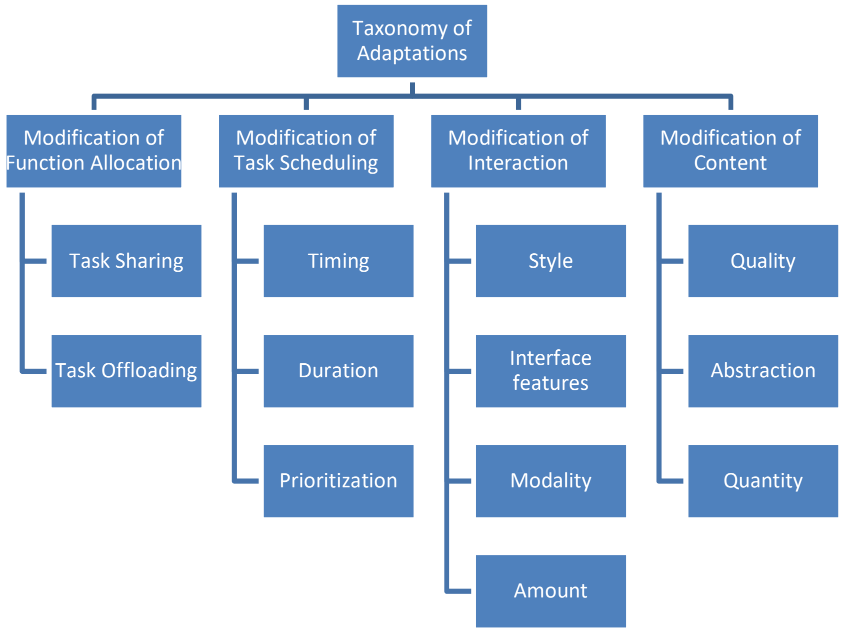


Figure 3: Taxonomy of Adaptations (Feigh et al, 2012)

Besides what is being changed, the trigger that initiates adaptivity is another essential part of adaptive systems. Feigh et al. (2012) developed a taxonomy of triggers (Figure 4), and distinguish five main categories:

1. Operator
2. System
3. Environment
4. Task/Mission
5. Spatio-Temporal

In this taxonomy, system refers to the control software managing the environment or plant, not to the e-partner. The environment represent the process(es) under control, of which their state or occurring event could instigate a trigger. The e-partner can use its models of the operator, system or environment to select appropriate triggers and the appropriate type of adaptation.

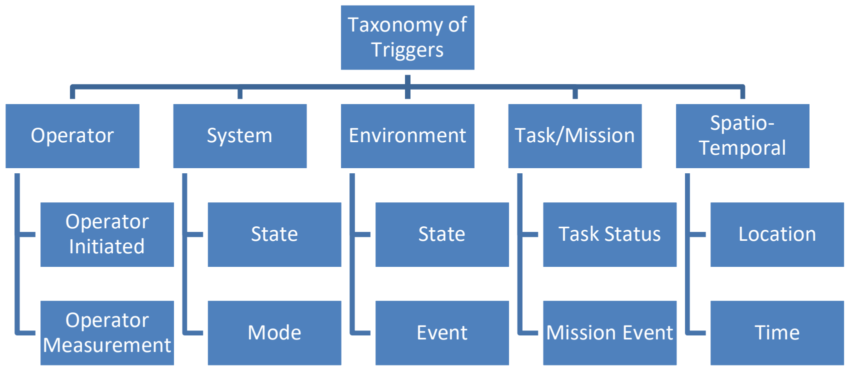


Figure 4: Taxonomy of Triggers (Feigh et al.,2012)

2.2 Supervisory displays

In our concept for the control of (semi-)autonomous systems, the ability for the operator to leave the control desk is one of the main goals. When not actively monitoring the systems, the operator can work on another desk, or could be walking around. This provides two different operational settings with different support opportunities.

In case the operator is sitting at a desk, supervisory information can still be presented in an easy manner. Inspiration can be found in work done by St. John and King (2010) on multi-tasking supervision and the development of the so-called Janus display (St. John, 2013). Their focus is multi-tasking supervisory support, where an operator has to monitor multiple highly-automated systems. They developed the four-second supervisor that enables the operator to detect deviant situations in one of its systems and turn to the primary task window in time.

Research conducted on peripheral displays could also provide concepts of interest, although they are normally designed in such a way that they are non-intrusive. Hence, they may be useful to signal non-urgent low risk changes in a situation (Matthews, Rattenbury, and Carter, 2007).

When the operator is not sitting at a desk but walking around and maybe even conducting physical tasks, opportunities for supervisory support are more limited. Information can be provided on mobile displays, but alerts may need to be generated to attract attention and the smaller screens have less space to provide detailed information. However, supervisory concepts similar to desk work should be usable.

Another field of research that may provide input to the development of supervisory displays is change blindness, a topic extensively studied. For example, Parasuraman, Cosenzo, and De Visser (2009) studied the monitoring and guiding of multiple UAVs with different levels of automation, presenting promising results on optimizing change detection for adaptive automation systems. A more thorough literature search must be conducted, as these topics where not part of the literature review in our previous report (Van der Kleij et al., 2015).

2.3 Skills, Rules, Knowledge (SRK) Framework

To understand the types of problems that are relevant for supervisory control systems, it is useful to distinguish between different types of behaviours. The Skill Rules Knowledge framework of Rasmussen (1983), distinguishes three categories:

- Skill-based behaviour represents behaviour that requires very little or no conscious control once an intention is formed. This is also known as a sensorimotor behaviour.
- Rule-based behaviour is characterized by the use of stored rules and procedures to select a course of action in a familiar work situation
- Knowledge-based behaviour represents reasoning at a higher conceptual level using an internal model of the system. This type of control must be employed when the situation is novel and unexpected.

The SRK framework has not only been successful in studying and understanding human behaviour, but also for understanding computer behaviours, and what they are good at. In short, computers perform very well at skill-based tasks, are capable of doing rule-based tasks, and have most difficulty at doing knowledge-based tasks. However, as the field of Artificial Intelligence progresses, more and more knowledge based tasks are also (partly) automated.

2.4 Just-in-time-awareness

Designing for SA is a topic extensively studied, for example by Endsley herself (Endsley and Jones, 2011). Endsley and Jones provide fifty SA design principles of which some may be useful for our work, but need further scrutinizing. For example, design principle number 34: "Automate only if necessary", clearly shows that in Endsley's model the operator is the holder of SA, whereas others propose a more system oriented view on SA (Stanton et al., 2006). This topic has recently been debated quite vigorously (Dekker, 2015; Endsley, 2015; Stanton, 2015). Because our environment consists of a human operator, an e-partner, and a control system, a distributed (system) perspective on SA is probably appropriate.

Most research addresses the maintenance of SA over prolonged periods. Little work seems to have been done specifically on the topic of fast SA development ore recovery (Gartenberg, Breslow, McCurry & Trafton, 2013; John & Smallman, 2008). Probably, if little time is available, an intuitive, naturalistic decision-making kind of approach will be taken, whereas when more time is available a rule-based or analytical knowledge-based approach may be chosen (Klein, 1993). This may provide indications of the type of support that is most beneficial to the operator.

3 Ontology

Before we can start implementing or specifying a prototype, the models as described in the previous section must be formalized and made consistent with each other. This section makes a step in this direction by defining an ontology: it describes all concepts that are needed to represent knowledge in this domain, and it does so in a uniform way. This ensures that knowledge can be shared between the different components of the system (e.g. actors, user interfaces, reasoning systems) because everybody speaks the same language.

3.1 What is an ontology?

An ontology is defined as a “formal specification of a shared conceptualization” (Gruber, 1993). This definition reveals two aspects which are important for our purposes. Firstly, as it is a formal specification, it can be used by a computer to represent and reason with knowledge. Most of the models about human behaviour lack that property, which means that they cannot be easily implemented, unless they are developed further into an ontology. Secondly, an ontology models a shared conceptualization. This means that they are used by multiple people and computers, which facilitates knowledge sharing as everybody speaks the same language. In this project, we aim to share the ontologies between the use cases for dynamic positioning, and (semi-)autonomous driving. Furthermore, the developed ontology will also be used within the defence research program Manning and Automation.

3.2 Ontologies for adaptive automation

To develop systems for adaptive automation, we need an ontology to represent knowledge in a computer and to be able to share information between different actors (human or machine) in the system. The subsections below describe our initial efforts in this area.

3.2.1 Top level ontology

The top level ontology is visualized in Figure 5. As this is only a top level ontology, the structure behind these classes is not yet specified and will be part of a later ontology-engineering effort. The important ontological commitment behind this model is that the context of a task must be modelled in three separate models for resources, environment and functions. For example the task *check-weather-forecast* serves to fulfil the function *Maintain-Situation-Awareness* in the environment *Sea* using the resources *DP-operator* and *DP-System*. Obviously this representation is too shallow to be useful for establishing adaptive task support in our domain. For example, it does not support reasoning over alternative resource allocations. However, it can be used for a first requirements analysis which in turn will help fleshing out this ontology in further depth.

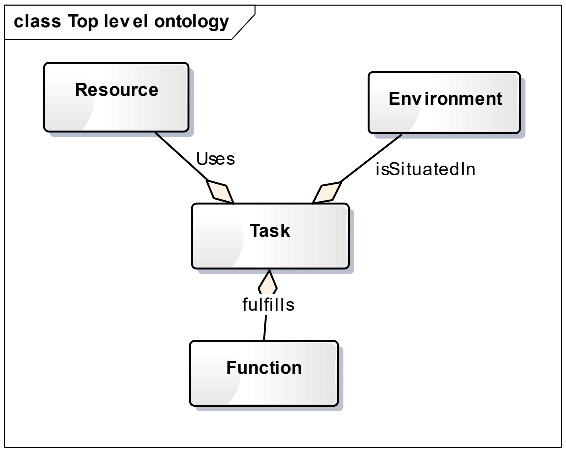


Figure 5 Top level ontology

3.2.2 *Resource ontology*

A resource is defined as anything that can be required by a task to enable task execution. This means that a resource can either be a consumable, a tool or a task performer. Different types of task performers exist: humans, but also machines. Humans can enact roles, which allow them to take tasks upon themselves. In the case of machines, we do not speak of roles, but of services. The resource ontology is depicted in Figure 6.

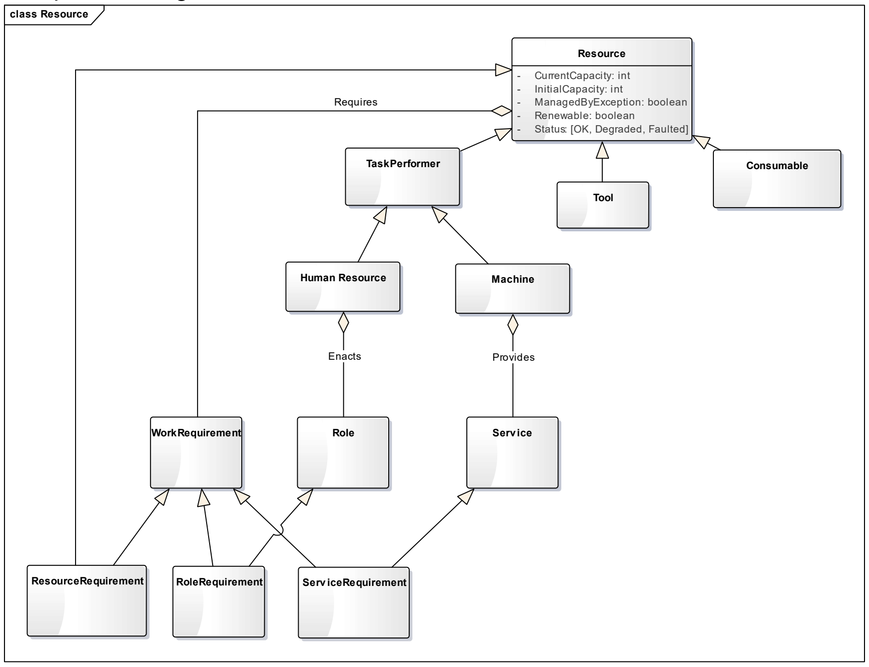


Figure 6: Resource ontology

3.2.3 *Task ontology*

The task ontology contains concepts that are required for specifying the demand of work. We distinguish among three different tasks:

1 tasks that have been observed in the past or in the present (descriptive tasks)
2 tasks that must occur according to regulations or plans (prescribed tasks)
3 tasks that are predicted to occur (predicted tasks).

The internal structure of a task is specified using workflows (i.e. tasks occur according to a specified control sequence, containing sequential composition, parallel composition and XOR splits). A specific type of activity is the "human activity", which are activities performed by humans. Activities performed by non-humans (e.g. agents) are described in the more general concept "Activity". Figure 7 presents the task ontology.

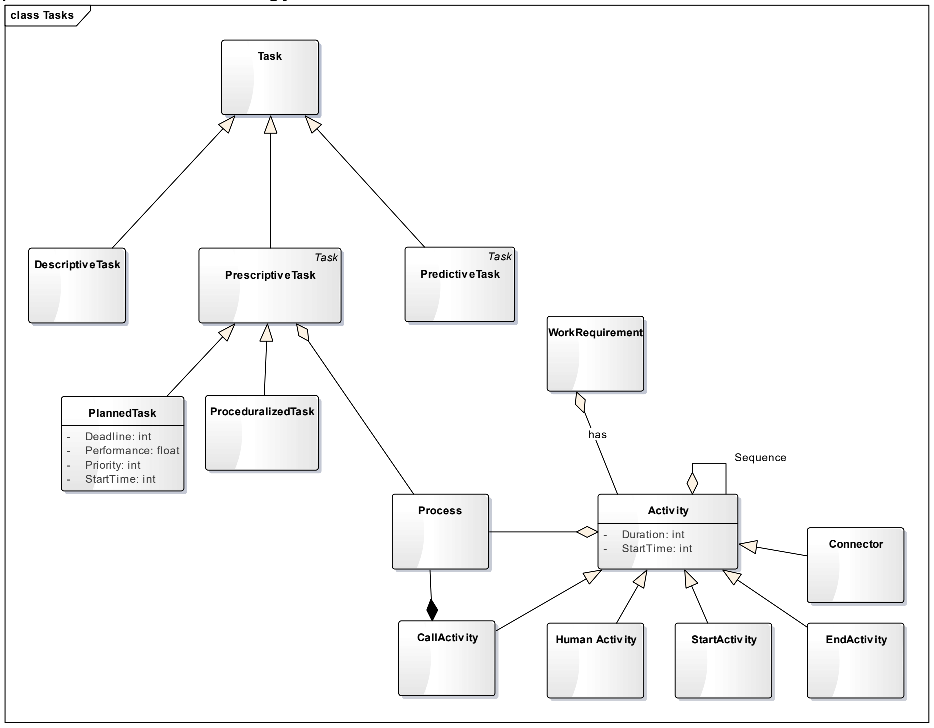


Figure 7: Task ontology

3.2.4 *Discussion*

This chapter presents an initial version of an ontology for adaptive automation. It focusses on four aspects that are considered important for adaptive automation, i.e. resources, functions, environment and tasks. This ontology is a useful basis for requirements engineering (which is the topic of the next section), as it adds meaning. For example, when a requirement states that the *status* of a *resource* must be monitored, this ontology defines that the status can be one of the values *OK*, *Degraded*, *Faulted*. This not only makes the requirement more precise, it also provides a basis for implementation, as the ontology can be translated straightforwardly into data structures of the implementation.

Obviously, this ontology is not set in stone and will change and be enhanced as research progresses. For example, during the implementation effort, it could turn out that the ontology is insufficiently expressive to allow for the kind of reasoning that is required. Therefore, this ontology should be regarded as a starting point which defines the concepts that are crucial for adaptive automation and which are usually left implicit in automated systems.

4 Concept Development and Experimentation

4.1 Situated Cognitive Engineering

To develop effective interactive, human-centred automation, as is our aim in this project, theory and empirical research should be built into the design process. To support the design processes systematically, a situated Cognitive Engineering (sCE) method was constructed. The sCE approach is based on the following development principles (Neerincx, 2012):

1. Creating human-centred automation is a multi-disciplinary collaborative activity
2. Functional modules are defined and tested incrementally in an iterative refinement process
3. Design decisions are explicitly based on claims analyses, explicating the up-downside trade-offs
4. Keeping and sharing the design rationale is key for progress and coherence in automation development

At the highest level, the sCE method distinguishes the Foundation, Specification and Evaluation phase (Figure 8). The Foundation contains the operational demands, human factors knowledge and technological principles. These give input to the Specification of a system at both the task and communication level of the system. This represents the concept development phase. Concepts developed are input for Evaluation, where experimentation takes place. The experimental results of the Evaluation will contribute to the knowledge base of the Foundation. We expect that part of this knowledge on adaptive automation concepts will be captured in design patterns that will be added to the Foundation for re-use.

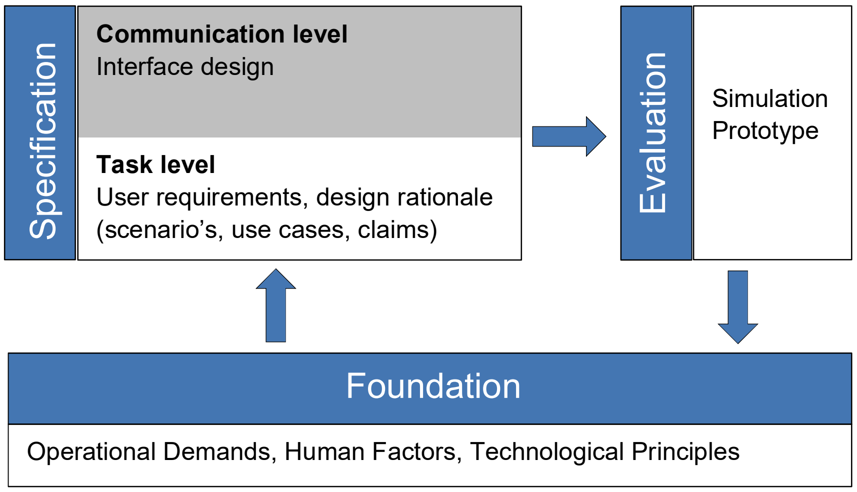

Specification
Communication level
Interface design
Task level
User requirements, design rationale
(scenario's, use cases, claims)
Evaluation
Simulation
Prototype
Foundation
Operational Demands, Human Factors, Technological Principles

Figure 8: Three main components of the sCE method (based on Neerincx, 2012)

4.2 Design patterns

A design pattern is a general reusable solution to a commonly occurring problem within a given context. Originally developed by an architect (Alexander, 1977) for urban planning and building design, the principle has become popular in software engineering (Gamma et al, 1993) and user interface design.

A design patterns provides a generic structure that, depending on context details, leads to a different but similar solution for different use cases. Many variations can be found in literature, but a pattern typically contains the following fields:

- **Pattern Name:** Choosing a clear and descriptive name helps people find the pattern and encourages clear communication between team members during design discussions.
- **Pattern Description:** Because short names like "one-window drilldown" are sometimes not sufficient to describe the pattern, a few additional lines of explanation (or a canonical screenshot) will help explain how the pattern works.
- **Problem Statement:** Written in user-centred language, this communicates what the user wants to achieve or what the challenge is to the end-user.
- **Use When:** "Context of use" is a critical component of the design pattern. This element helps people understand situations when the design pattern applies (and when it does not.)
- **Solution:** The solution should explain "how" to solve the problem, and may include prescriptive checklists, screenshots, or even short videos demonstrating the pattern in action.
- **Rationale:** Providing reasons "why" the pattern works will reinforce the solution, though time-pressed developers may prefer to ignore this explanation.
- **Examples:** Each example shows how the pattern has been successfully applied
- **Comments:** Including a place for team members to discuss the use of the pattern helps maintain an active resource and keeps the team engaged.

Design patterns provide a powerful and practical method to capture knowledge developed into reusable packages that can be tweaked to similar problems. Most work on design patterns and adaptivity is focused on adaptation of the user interface. For example, Peissner and Edlin-White (2013) have studied adaptation based on personal characteristics, such as lower visual or auditory perception recognition ability. Larger fonts or a different use of sounds are chosen automatically by the system. Hence, referring to the taxonomy of Feigh et al. (2012), this work focusses on the modification of interaction and the modification of content.

Peissner and Edlin-White (2013) also defined two adaptation dialogue patterns. In the first, the operator is informed about the adaptation, and has the option to undo this process. The second pattern describes an explicit conformation pattern, which means the operator has to accept the adaptation explicitly before or after the adaptation is executed. The patterns are evaluated based on controllability and transparency. It was concluded that complex systems will need both, as each has its own advantages and costs.

Design patterns developed will be added to the Foundation as described in the previous section on sCE.

4.3 Designing for adaptivity in human-automation systems

Johnson, Bradshaw, Feltovich, Jonker, Riemsdijk and van Sierhuis (2011) present a method called Coactive Design that can be used by developers to translate high-level teamwork concepts into control algorithms, interface elements, and behaviours that enable robots to fulfil their envisioned role as teammates. They developed a human-robot system model that supports collaboration through attention to requirements for observability, predictability, and directability. These categories can be traced to the 10 “automation as a team player” principles put forth by Klein et al. (2004) and are defined as follows (Johnson et al., 2011):

- Observability means making pertinent aspects of one’s status, as well as one’s knowledge of the team, task, and environment observable to others. Since interdependence is about complementary relations, observability also involves the ability to observe and interpret pertinent signals. Observability plays a role in many teamwork patterns e.g., monitoring progress and providing backup behaviour.
- Predictability means one’s actions should be predictable enough that others can reasonably rely on them when considering their own actions. The complementary relationship is considering others’ actions when developing one’s own. Predictability is also essential to many teamwork patterns such as synchronizing actions and achieving efficiency in team performance.
- Directability means one’s ability to direct the behaviour of others and complementarily be directed by others (others can be human or machine actors). Directability includes explicit commands such as task allocation and role assignment as well as subtler influences, such as providing guidance or suggestions or even providing salient information that is anticipated to alter behaviour, such as a warning. Teamwork patterns that involve directability include such things as requesting assistance and querying for input during decision making.

Design patterns developed in this study will take these three dimensions into account to optimize for transparency in joint-control.

4.4 Experimental platform

An experimental platform will be developed that facilitates the experimentation and validation of automation concepts developed in this project. This platform will be described in more detail in the next chapter..

5 Use case: Dynamic Positioning

5.1 Introduction

Our use case focuses on the stationary dynamic positioning (DP) operator at Floating Production, Storage and Offloading (FPSO) platforms (see Figure 9). For more information on FPSOs, please refer to Van der Kleij et al. (2015). Stationary DP operations are a typical example of a control task that is highly automated but does require human supervision. Most of the time, the operator monitors the system but does not have to take action. It is quite rare that system failure or environmental circumstances require human intervention. However, when an intervention is required the operator has to be fast, as generally very little time is available before dangerous situations occur. Quickly understanding what is going on followed by appropriate action taking is a complex task that operators sometimes fail at, resulting in environmental spills, life-threatening situations, or large economic damage.

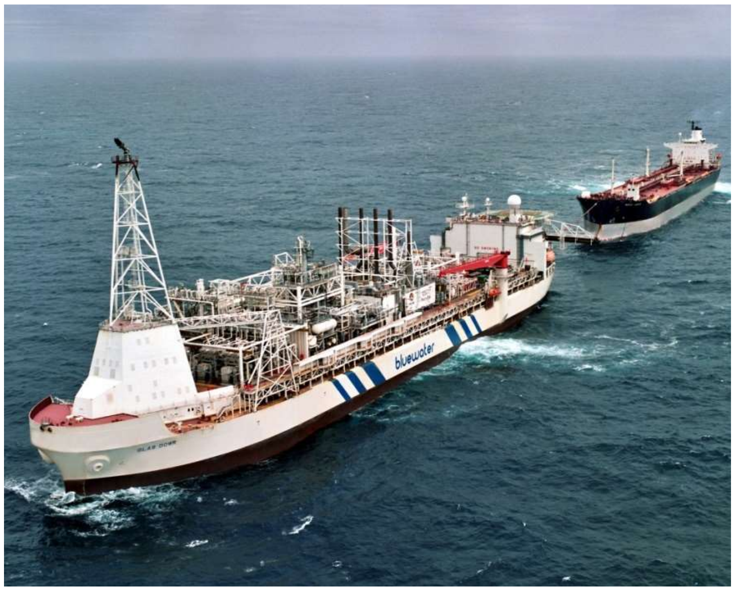


Figure 9: FPSO on DP during offloading

Figure 10 shows the future DP control environment we envision. An FPSO is managed by a DP system consisting of control software, sensors and thrusters. The DP software can be monitored and managed by either the DP operator or by an e-partner. This is an addition to current practice, where this type of support is non-existent.

As described in paragraph 2.2, the aim of our support concept is to enable the operator to leave the DP desk during stable periods of low risk to conduct other activities. He may get regular status updates (notifications) by the e-partner, and when required can return to the desk and take over active control. In this case, the e-partner supports the quick development of situation and option awareness. Hence, the e-partner has two main tasks:

- Proactively support the roaming operator with updates whenever relevant, to keep the operator at a minimally required level of SA.
- When required, actively support the DPO with just-in-time awareness when he has to take over control immediately

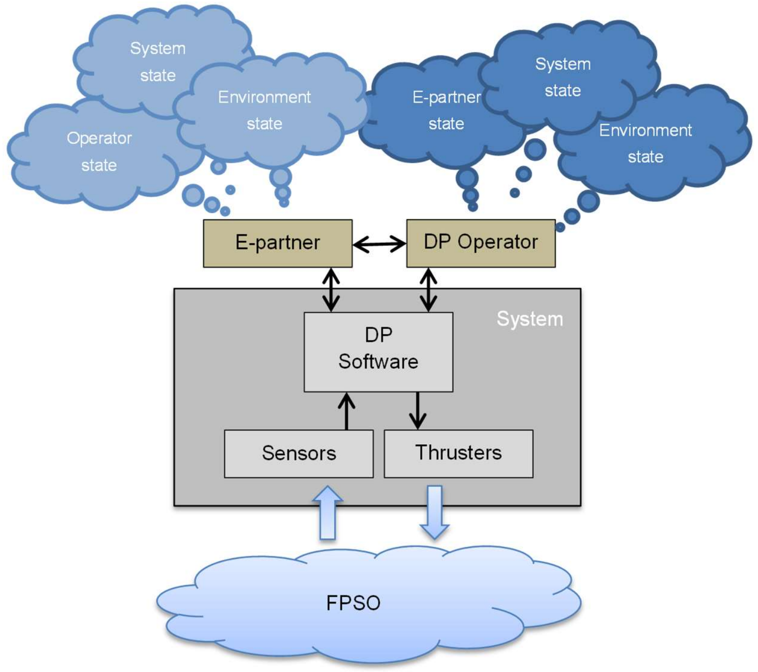


Figure 10: The high level concept of a control environment for future DP operators

The next two paragraphs will address the research questions and propose solutions that will be implemented and evaluated.

5.2 Maintaining minimal SA: the use of smart notifications

Modern SCADA (Supervisory Control And Data Acquisition) systems, such as DP systems, can be regarded as having two control loops. The first loop is a fully autonomous loop where the system responds to triggers in the environment. As an (oversimplified) example, if the ship on DP is drifting eastwards, the power on the thrusters facing eastwards is increased. The second control loop serves to deal with those situations in which the computer cannot establish a solution autonomously. In this case, the computer recognizes a problem, alarms the user, and then passes full control to the user. If we map these control loops to the Skill-Rule-Knowledge framework discussed in Section 2.3, we observe that the computer is applied to perform skill-based and rule-based behaviours. The skill-based behaviours are implemented in the first control loop, where sensor data are immediately mapped to action. Rule-based behaviours are implemented in the second control loop, where rules are used to trigger alarms that are followed up by users who follow procedures to deal with these alarms.

Currently, the knowledge-based control loop is not supported in SCADA applications. The reason for this is not that Artificial Intelligence is not capable to reason or assist with those types of problems. Today, many examples exist of operational systems that assist in knowledge based solutions using advanced symbolic reasoning systems, machine learning, and big data analytics techniques (e.g. Neerincx, 2011). (One of) the reasons that these solutions are not frequently applied in SCADA systems, is that the current HMI paradigm, which is based on alarms, does not suffice. Alarms are a very shallow way of communicating information to the user. This works well when the alarm is triggered by a simple and non-controversial fact (for example, when a threshold is exceeded), as in skill-based and rule-based problems. In knowledge-based problems, the trigger may be less concrete (e.g., a weak signal), and it may be unclear what the proper response should be: who should resolve the problem (computer or human or both)?, is it a problem at all (the computer may be wrong, or the operator may be wrong)?, when does the problem require a response (now, or can it be postponed)? For this reason, we believe that an additional layer of communication should be introduced that supports knowledge-based problem solving. We call this type of communication 'smart notification' (Figure 11).

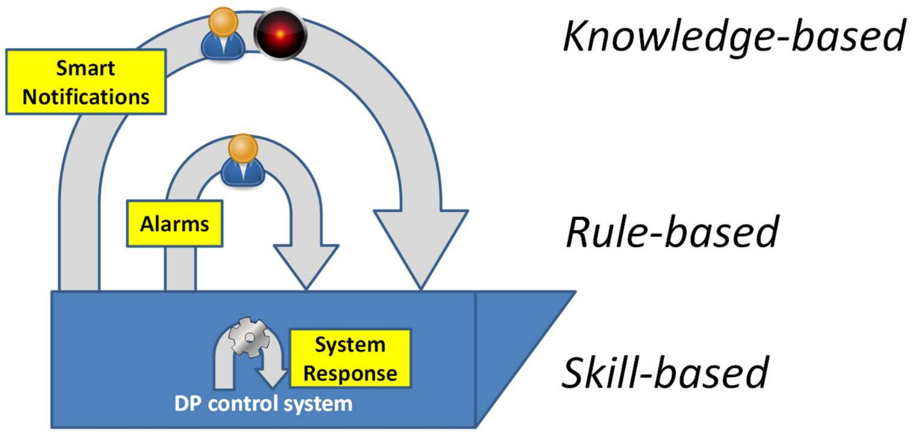


Figure 11: Introducing an additional control loop "smart notifications" in SCADA applications

5.2.1 *Difference between alarms and smart notifications*

To understand to concept of smart notifications, it is useful to distinguish them from alarms. The definition of an alarm is as follows (according to ISA 2009):
An audible and/or visible means of indicating to the operator an equipment malfunction, process deviation, or abnormal condition requiring a response.

This definition defines four important characteristics of alarms (DeltaV 2013):

1 There must be an indication of the alarm (audible or visible). An alarm limit can be configured to generate control actions or log data without it being an alarm.
2 The indication must be targeted to the operator to be an alarm, not to provide information to an engineer, maintenance technician, or manager.
3 The alarm must indicate a problem, not a normal process condition.
4 There must be a defined operator response to correct the condition. If there is no operator response necessary, then there should not be an alarm.

The main reason why alarms are not suitable to use as a basis for smart notifications, is that knowledge-based decisions (by definition) have no predefined operator response to correct the condition.

The ISA standard also defines another concept, called an alert, which is similar to an alarm, but which has a somewhat weaker definition:
An audible and/or visible means of indicating to the operator an equipment or process condition that requires awareness, that is indicated separately from alarm indications, and which does not meet the criteria for an alarm.

However, the definition of an alert is still not sufficient to use as a basis for applying smart notifications, because the responsibility for dealing with the alarm or alert is still transferred to the operator. In the human-machine team paradigm, many more options exist, such as shared responsibility or collective obligations (Van Diggelen, 2010). To capture these aspects, smart notifications can be applied, which do not simply push problems to the operator, but builds up mutual collaborations and are dedicated to solve problems that arise. The differences between alarms, alerts, and smart notifications are outlined in Table 1.

Table 1: Alarms, alerts and smart notifications

	Alarms	**Alerts**	**Smart Notification**
Responsibility for handling the alarm or notification	Operator	-	Operator and Intelligent Operator Support System
Trigger	Simple (e.g. threshold, or rule-based)	Simple	Complex (e.g., resulting from ML or data fusion of heterogeneous sources)
Communication type	One shot	One shot	Dialogue
Modality	Short text	Short text	Text, pictures, graphs, movies, sounds.
Content	Facts	Facts	Facts, judgements, predictions (which may be unreliable), contextual information.
Intended effect	Action by operator	Awareness by operator	1) Action by operator-SOS team. 2) Negotiation to reach consensus on how to proceed

Whereas the concept of Smart notifications allows for establishing a more advanced human computer interaction within SCADA systems, it still requires very good design practices to actually realize them. These design practices will be laid down in design patterns.

5.2.2 *Proposed Design patterns*

To design a good smart notification, many decisions must be made on aspects such as:

- What is the modality? If pictures are used, what do these look like? If sonification is used, what type of signal should be chosen?
- How is the dialogue structured? What are the response options for the operator? What are the answers of the computer?
- Who takes the initiative? Does the system proactively start communications with the operator, or is it only reactive? If so, when does it do so?
- How does operator/system state affect the smart notifications? If the operator is busy, does the smart notification respond to that and become less occupying?
- How do the different smart notifications behave with respect to each other? Are the most important ones shown more prominently? What happens to smart notifications that are no longer relevant?

Much work remains to be done to design these in a proper way, and to evaluate them. The current project has just scratched the surface. Below, two proposals for design patterns are shown that describe two types of smart notifications. Note that this format is more light-weight than the format we originally proposed in Section 4.2.The remaining pattern fields can be described after evaluation.

The first design pattern (Table 2) aims at overcoming a problem present in most modern SCADA systems (Supervisory Control And Data Acquisition), where either the computer has full autonomy, or where the full control is passed to the human via an alarm. Using this design pattern, a novel way of interaction is introduced which is a combination of alarming and supervisory controlled automation.

Table 2 Design pattern for Request approval for corrective action

Pattern Name	Request approval for corrective action
Pattern Description	The computer notices a problem (could be an alarm) and suggests a solution that it will execute if the human approves.
Problem Statement	- Operator needs sufficient understanding of the problem to be able to judge the solution. - Explaining the solution should not take too much time
Use When	An automated solution exists, but is too unreliable to be implemented without human supervision.
Solution description	Popup window with short explanation of the problem and proposed system solution. The user can ask the system for more explanation, and decide to approve or disapprove the proposed solution.

The second design pattern aims at providing a solution for the problem that human control has some context requirements which must be fulfilled before control can be passed to the human (Table 3). One of these context requirements is spatial location. For example, when the system operates in fully autonomous mode, and a problem occurs, the human operator should be able to make it back to the workstation within a certain time limit.

Table 3: Design pattern for Demand operator to stay in vicinity

Pattern Name	Demand operator to stay in vicinity of workstation.
Pattern Description	The computer that autonomously executes a task predicts that human intervention might be necessary soon, which cannot be done from a mobile device. It asks the operator to stay in the vicinity of the workstation.
Problem Statement	- Operator needs understanding why this is necessary. - Computer needs feedback if the operator approved. - Computer needs to know operator location
Use When	Computer expects to switch from autonomous mode to a semi-autonomous mode which requires a stationary operator.
Solution description	Popup window with short explanation of the type of expected problems and time frame. The user can ask the system for more explanation, and decide to agree or disagree to stay in the vicinity.

5.3 Just in time awareness

Whereas initial solutions have been developed for smart notification, work has yet to start on just-in-time-awareness concepts. Based on the lines of research introduced in this document, various concepts will be developed, implemented in our demonstrating environment and evaluated in line with the sCE method.

5.4 Demonstrator

We have developed a simple demonstrator to demonstrate the idea of smart notifications. The architecture of this demonstrator is shown below:

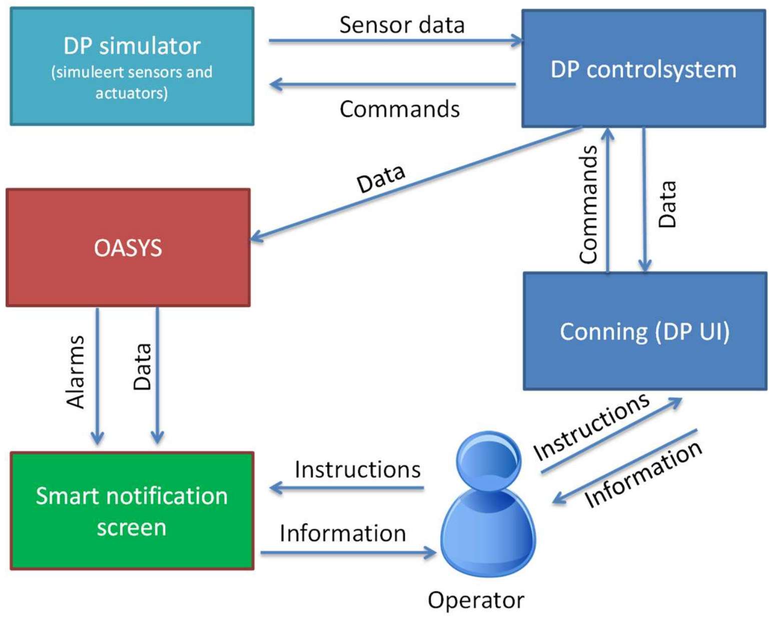


Figure 12: System architecture of the Demonstrator

The backbone (i.e., the components of the system that are not visible for the human operator) of this architecture are made up of:

- The DP simulator. This component is developed by RH Marine (formerly known as IMTECH), and simulates the ship. In operational conditions, this component would be absent, and be replaced by a real vessel.
- The DP control system is the system that reads sensor data, computes what the proper response should be, and subsequently controls the actuators. This component is also developed by RH Marine. It comes with a front end interface called Conning.
- OASYS is a layer on top of the control system that generates additional alarms, and allows for easy prototyping of alarm-based systems. In this demonstrator, only the alarm-generation functionality was used, the user interface functionality of this component was not used. This component is developed by the company UReason.

The front end (i.e. the components of the system that are visible to the user) are the Conning user interface and the Smart Notifications interface. Both of these interfaces will be described in more detail below.

5.4.1.1 *Conning interface*

The conning interface is developed by RH marine, and is the interface as it is currently used by DP operators to monitor sensor values, to direct actuators, and to set the inputs for the control system (such as positioning location). Figure 13 shows the main view of this interface.

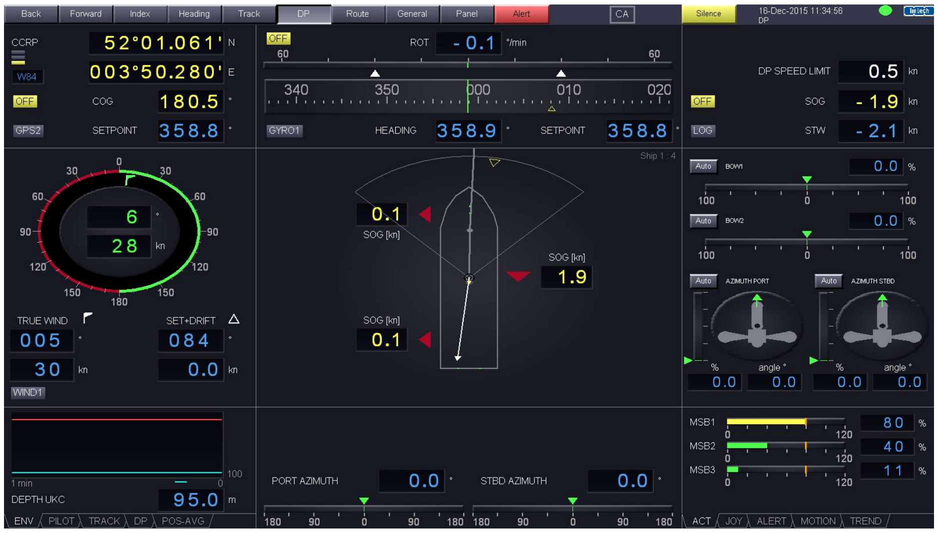
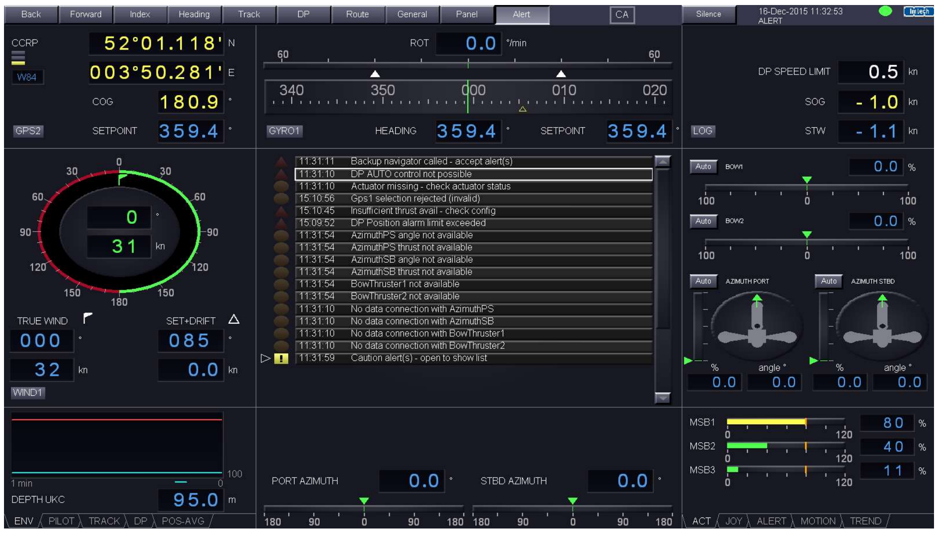


Figure 13: Conning DP interface

Figure 14 shows the alarm screen of the Conning DP interface, showing the alarms as a long list which in case of incidents or failure easily overwhelms operators without providing structured information that helps to develop a deeper understanding of what is going on.

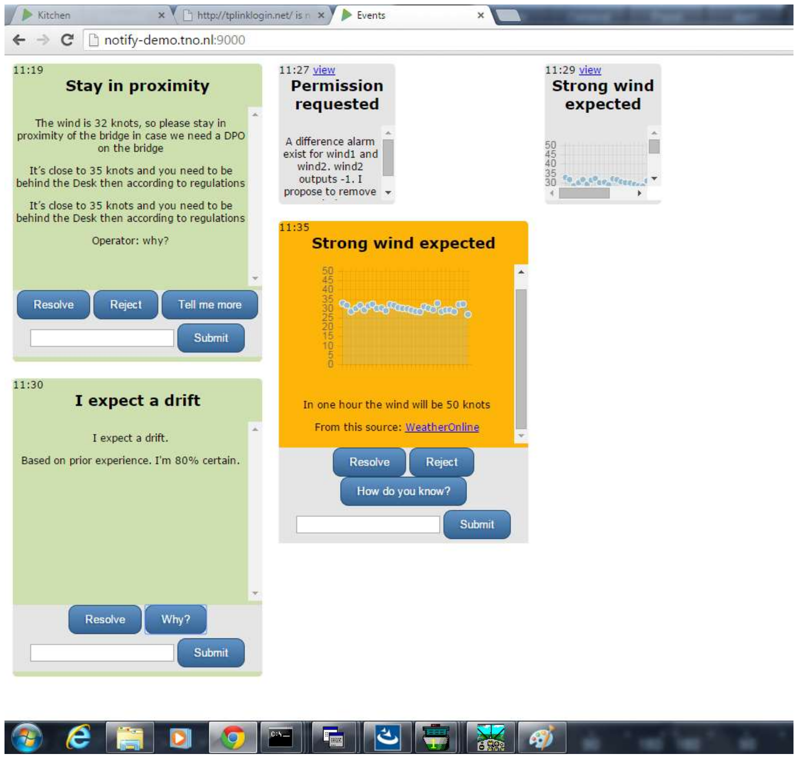


```
11:31:11  Backup navigator called - accept alert(s)
11:31:10  DP AUTO control not possible
11:31:10  Actuator missing - check actuator status
15:10:56  Gps1 selection rejected (invalid)
15:10:45  Insufficient thrust avail - check config
15:09:52  DP Position alarm limit exceeded
11:31:54  AzimuthPS angle not available
11:31:54  AzimuthPS thrust not available
11:31:54  AzimuthSB angle not available
11:31:54  AzimuthSB thrust not available
11:31:54  BowThruster1 not available
11:31:54  BowThruster2 not available
11:31:10  No data connection with AzimuthPS
11:31:10  No data connection with AzimuthSB
11:31:10  No data connection with BowThruster1
11:31:10  No data connection with BowThruster2
11:31:59  Caution alert(s) - open to show list
```

Figure 14: The alarm screen of the Conning interface.

5.4.1.2 *The smart notification screen*

The smart notifications screen is shown in Figure 15. The coloured tiles/windows contain active smart notifications. The grey windows contain inactive smart notifications. We will briefly describe the active notifications below



11:19
Stay in proximity

The wind is 32 knots, so please stay in proximity of the bridge in case we need a DPO on the bridge

It's close to 35 knots and you need to be behind the Desk then according to regulations

It's close to 35 knots and you need to be behind the Desk then according to regulations

Operator: why?

11:27 view
Permission requested

A difference alarm exist for wind1 and wind2. wind2 outputs -1. I propose to remove

11:29 view
Strong wind expected

11:35
Strong wind expected

In one hour the wind will be 50 knots

From this source: WeatherOnline

11:30
I expect a drift

I expect a drift.

Based on prior experience. I'm 80% certain.

Figure 15: Smart Notifications screen

The green notification at the upper left corner contains a warning that the user should stay in the proximity. After the user had asked “why?”, the system has given more information. If the user would choose to resolve the issue, the window would turn grey.

The green notification at the lower left corner contains a notification that stems from a machine learning algorithm. This means that the system has derived that the probability of a drift is high, using, for example, an artificial neural network. This means that the system cannot explain why this is the case (it can just give a probability of its error rate). Depending on the *trust* that the operator has in this system, he/she can choose to ignore this advice or to follow it. This is also one of the reasons why such a separate smart notification screen is important: to allow the user to distinguish between factual information on the conning information, and judgemental information on the smart notification screen (i.e., information which may be wrong).

The yellow window contains a smart notification that uses a graph to communicate information. In this case, the graph contains the wind speed. Based on the trends observed by the user, the user can decide if he/she finds the situation alarming or not.

To be able to evaluate the ideas described above, we will formulate them as design patterns, and evaluate them in an experiment with test subjects.

6 Conclusions

Next generation human-automation collaboration will incorporate flexible and adaptive collaboration styles, especially when highly autonomous systems are involved. Smart notification concepts combined with just-in-time awareness support can provide new types of work agreements that enable operators to conduct other tasks during quiet or safe periods, and engage the operator when things get difficult. Intelligent e-partners with a thought understanding of the processes under control, the automated controlling system, and the operator, play a central role in these new concepts.

A CD&E research framework consisting of a design methodology (sCE), novel concepts to support operators of highly autonomous systems with multi-tasking (smart notifications and just-in-time awareness), and a platform for experimentation and demonstration (DP simulator with Conning interface and OASIS software) was presented. First ideas on smart notifications have been implemented and demonstrated, giving us hands-on experience with the different parts of the platform and its viability for experimentation and demonstration purposes.

In 2016, innovative smart notifications and solutions to quickly bring operators back in the loop will be developed and evaluated on the experimentation platform. In an iterative manner, more sophistication will be added and validated.

We expect the concepts developed and validated on the DP experimentation platform to be easily transferable to other domains, as the problems tackled are generic by nature. Autonomous sailing is an obvious candidate in the maritime domain, but highly automated processes in other domains (navy, mobility, utility management) are likely to provide ample opportunities for application of our concepts.

7 References

International Society of Automation (2009), Management of alarm systems for the process industries, ANSI/ISA–18.2–2009, https://www.isa.org/templates/one-column.aspx?pageid=111294&productId=116626

Alexander, C. (1977). A pattern language: towns, buildings, construction. *Oxford University Press*, USA. p. 1216. ISBN 0-19-501919-9.

Dekker, S.W.A. (2015) The danger of losing situation awareness. *Cogn Tech Work* 17:159–161

DeltaV, Alarm Rationalization (2013), DeltaV Whitepaper, http://www2.emersonprocess.com/siteadmincenter/PM%20DeltaV%20Documents/Whitepapers/WP_Alarm_Rationalization.pdf

Van Diggelen, J., Bradshaw, J. M., Johnson, M., Uszok, A., & Feltovich, P. J. (2010). Implementing collective obligations in human-agent teams using KAoS policies. In C*oordination, Organizations, Institutions and Norms in Agent Systems V* (pp. 36-52). Springer Berlin Heidelberg.

Endsley, M. R. (1995). Toward a theory of situation awareness in dynamic systems. *Human Factors,* 37, 32–64.

Endsley, M. R. (2015). Situation awareness misconceptions and misunderstandings. *Journal of Cognitive Engineering and Decision Making*, 9(1), 4–32.

Endsley, M. R., & Jones, D. G. (2012). Designing for situation awareness: an approach to human-centered design (2nd ed.). London: Taylor & Francis.

Feigh K.M., Dorneich, M.C. and Hayes, C.C. (2012) Toward a characterization of adaptive systems: a framework for researchers and system designers. *Human Factors*, 54(6):1008-24.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1993). Design patterns: abstraction and reuse of object-oriented design (pp. 406-431). Springer Berlin Heidelberg.

Gartenberg, D., Breslow, L., McCurry, J. M., & Trafton, J. G. (2013). Situation awareness recovery. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 0018720813506223

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2), 199-220.

John, M. S., & Smallman, H. S. (2008). Staying up to speed: Four design principles for maintaining and recovering situation awareness. *Journal of cognitive engineering and decision making*, *2*(2), 118-139.

Johnson, M., Bradshaw, J.M., Feltovich, P.J., Jonker, C.M., Riemsdijk, M. B. van and Sierhuis, M., (2014) Coactive design: designing support for interdependence in joint activity. *Journal of Human-Robot Interaction*, Vol. 3, No. 1, 2014, pp. 43-69.

Kleij, R. van der, Broek, H. van den, Brake, G.M. te, Rypkema, J., Schilder, C. (2015) ERP-HE AMA Progress Report: Use Case and Computational Model for Adaptive Automation. TNO report.

Klein, G. A. (1993). A recognition-primed decision (RPD) model of rapid decision making (pp. 138-147). Ablex Publishing Corporation.

Klein, Gary, David D. Woods, J. M. Bradshaw, Robert Hoffman, and Paul Feltovich. (2004) “Ten challenges for making automation a “team player” in joint human-agent activity.” *IEEE Intelligent Systems* 19, no. 6: 91-95.

Matthews, T., Rattenbury, T., & Carter, S. (2007). Defining, designing, and evaluating peripheral displays: An analysis using activity theory. *Human–Computer Interaction*, 22(1-2), 221-261.

Neerincx, M.A. (2011) Situated cognitive engineering for crew support in space. *Personal and Ubiquitous Computing*. Volume 15, Issue 5, pp. 445-456.

Parasuraman, R., Cosenzo, K.A. and De Visser, E. (2009) Adaptive Automation for Human Supervision of Multiple Uninhabited Vehicles: Effects on Change Detection, Situation Awareness, and Mental Workload, *Military Psychology*,21:2,270 — 297

Peissner, M., Edlin-White, R. (2013) User control in adaptive user interfaces for accessibility, *Human-Computer Interaction – INTERACT* 2013. Volume 8117 of the series Lecture Notes in Computer Science, pp 623-640

Rasmussen, J. (1983). Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. *Systems, Man and Cybernetics, IEEE Transactions on*, (3), 257-266.

Sheridan, T. B. (2011). Adaptive automation, level of automation, allocation authority, supervisory control, and adaptive control: Distinctions and modes of adaptation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(4), 662-667.

Sheridan, T. B. (2012). Human supervisory control, in *Handbook of Human Factors and Ergonomics,* Fourth Edition (ed G. Salvendy), John Wiley & Sons, Inc., Hoboken, NJ, USA. doi: 10.1002/9781118131350.ch34

Stanton N.A., Stewart R, Harris D, Houghton R.J., Baber C, McMaster, R, Salmon P.M., Hoyle G., Walker G.H., Young M.S., Linsell M., Dymott R, Green, D. (2006) Distributed situation awareness in dynamic systems: theoretical development and application of an ergonomics methodology. *Ergonomics* 49:1288–1311

Stanton, N. A., Salmon, P. M., & Walker, G. H. (2014). Let the reader decide: a paradigm shift for situation awareness in sociotechnical systems. *Journal of Cognitive Engineering and Decision Making,* 1555343414552297.

St. John, M.F. (2013). Janus Design Principles and the Acquisition Process for a Supervisory Situation Awareness Display. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (Vol. 57, No. 1, pp. 2002-2006). SAGE Publications.

St. John, M. F. S., & King, M. A. (2010, September). The four-second supervisor: multi-tasking supervision and its support. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (Vol. 54, No. 4, pp. 468-472). SAGE Publications.