Requirement Elicitation and Validation by Prototyping and Demonstrators
User Interface Development in the Oil- and Gas Industry

Jan Magnus Røkke
Dresser-Rand AS
Granittveien 3C
3615, Kongsberg, Norway
magnus.roekke@live.no

Gerrit Muller
Buskerud university College
Kongsberg, Norway

Michael Pennotti
Stevens Institute of Technology
Hoboken, NJ, USA

Abstract

Incomplete or misinterpreted requirements are a significant source of customer and user dissatisfaction in development of software user interfaces. In these systems, where consideration of the human factor is a vital part of the development, the undertaking of understanding the real needs of the user must not be underestimated. Unfortunately, there are often organizational boundaries which restrict or limit the developer's opportunities to communicate with the customer and stakeholders. The result is often a weak link between the stakeholder needs, system requirements and the realization of the user interface system.

This paper addresses how an approach to requirements engineering based on a combination of rapid prototyping and demonstrator sessions can be used to elicit requirements and obtain early feedback and acceptance from system stakeholders. The method was conducted on a user interface development project for gas turbine driven generator- and compressor packages in operation at offshore oil-rigs. Stakeholders were presented with module prototypes with a varying degree of dynamics, simulation and interaction based on the stage of the development. Together with rationale based questioning, the demonstrator sessions provided a context for constructive discussions and feedback. The developers returned with a better understanding of the rationale for stakeholder need and clarification of misinterpreted or poorly defined requirements. This enabled us to create an application better aligned with customer and user needs and a minimal amount of rework and updates after system deployment.

1 Introduction

1.1 Requirement Feedback & Validation

Requirements engineering is a well established domain in software development. It involves finding out what people want from a system and translating their needs into design specifications. The requirements must be validated to ensure that the right system will be created. At the end of the development the system is validated to ensure it fits the acceptance criteria. If the system fails at the end of the development phase to pass the acceptance criteria then the outcome may be project failure or at least huge expenses to correct for the missing or misinterpreted requirements. The cause of missing or misinterpreted requirements might be the lack of thorough requirements engineering in the development phase leading to emergent needs after system commissioning.

One of the most important means to achieve a successful product is the abundant use of feedback from the project stakeholders. (Muller, 2008) There are several methods for obtaining feedback on system requirements. The most common approach is presenting the customer with a set of documents like a requirements specification. However, a challenge facing engineers is the level of formality used in these documents. They are often full of lists and tables defining requirements and detailed system specifics. As customer and stakeholders may not be familiar with computer notations and technical language they might have a challenge in interpreting these documents in terms of what they really want. If they are not able to visualize the proposed solution they can have a hard time providing the developer with feedback uncovering their real needs. Alternative methods for feedback employing a more informal context and natural language should be considered. This paper studies the use of prototypes and demonstrators as a method for early feedback and validation of requirement and design specifications. In the paper we will show how the method was applied from the early stages of a project where the developed product is highly interactive.

1.2 Case Background

Dresser-Rand [DR] is a global company delivering gas turbine application solutions to oil and gas production, refining, gas processing, storage and transportation and other general industry. Their activities includes design, manufacturing and servicing a wide range of technologically advanced centrifugal compressors, steam and gas turbines, expanders, multiphase turbine separators and control systems. One of Dresser-Rand's branches is the Norwegian operation, located in Kongsberg. Dresser-Rand Kongsberg’s major area of business lies in packaging gas turbines from General Electric and Rolls Royce driving both compressors and generators for energy industry applications. The two main applications are compression of gas for long range transport and generation of power for off-shore and land based installations. The packaging of these systems also includes in-house manufacturing of customized control systems, acceptance testing and service installation.

Due to an expanding global market and stronger competition in the area, Dresser-Rand Kongsberg is focusing on development methods and design choices that could benefit the company in the long run. This includes exploring various SE [Systems Engineering] techniques and tools and their value in processes surrounding the development of gas turbine systems.

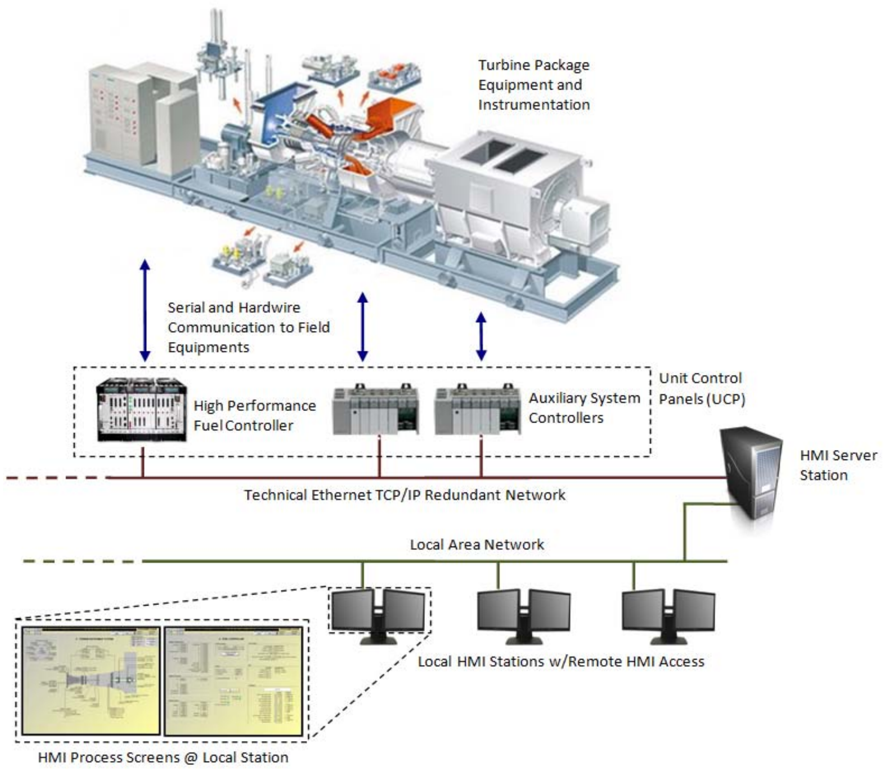
Turbine Package Equipment and Instrumentation
Serial and Hardwire Communication to Field Equipments
Unit Control Panels (UCP)
High Performance Fuel Controller
Auxiliary System Controllers
HMI Server Station
Technical Ethernet TCP/IP Redundant Network
Local Area Network
Local HMI Stations w/Remote HMI Access
HMI Process Screens @ Local Station

Figure 1: Gas turbine control system topology

The case study at Dresser-Rand dealt with the HMI [Human-Machine-Interface][1] system, also called GUI [Graphical-User-Interface]. In the study we investigated how the development process and the HMI product could be improved by help of rapid prototyping and demonstrators for early feedback and validation of requirements and design choices. The methods were applied on a single project for a larger customer of Dresser-Rand, and the success of the method caused an increase of the scope of the

[1] Human-Machine Interface is the aggregate of means by which people—the users—interact with the system—a particular machine, device, computer program or other complex tool.

application. In this paper the author will describe the motivation for the use of demonstrators as a tool in the development phase of the HMI system and how the SE process and demonstrators were carried out. The paper will show how the demonstrators were received by the stakeholders and the result of employing the method in the development. Finally there are some suggestions for where and how this method can be employed in other industries as well as suggestions for further research into this area.

1.3 Case Introduction

The generator and compressor applications are high speed processes controlled by high performance fuel controllers, controlling the turbine itself by managing the flow and distribution of fuel to the turbine, as well as the auxiliary system PLCs [Programmable Logic Controller] controlling lube oil systems, ventilation, safety system and so forth. Interfaced to these controllers, as depicted in Figure 1, is the human-machine-interface system which has the capability to communicate with one or multiple PLCs and other types of controllers. The communication occurs over an Ethernet TCP/IP link using multiple forms of communication protocols. Process data can be collected and archived on servers to facilitate further remote analysis and sharing. The HMI is a tool for the operator to monitor the current conditions of analog and discrete inputs, timers, alarms and shutdowns, as well as other operational status information. It provides alerts and alarms when the process is approaching dangerous conditions, as well as giving the operator a method of starting, stopping and controlling the machinery and processes. It gathers records which are archived to facilitate the generation of reports and to allow historical data to be reviewed at a later date. The operator can also trend data gathered over long periods to highlight slow changes or to review causes of past process disturbances.

Even though the HMI is the primary tool by which an operator can view the process and machinery being controlled, it does not perform any logic or calculations directly controlling the process or alarm treatment. The PLCs perform the real-time, critical, control and the HMI performs the monitoring (reflecting the states of process values and alarms handled by the PLC), and is used for calibration and manual control of equipment when not in normal operation. As such the HMI does not have a high criticality with regards to the turbine application, which might be one of the reasons it has not been prioritized by any of the involved parties during project development.

However, in later years there has been an increased attention to industrial human-machine-interfaces systems, both within Dresser-Rand and the rest of the industry. As a part of the process of addressing the various issues we are facing when it comes to our control systems, Dresser-Rand wanted to investigate how we could develop our HMI systems with a minimal amount of rework and fixes after FAT [Factory-Acceptance-Test] and commissioning.

1.4 Problem Statement

As a provider of engineering solutions for industrial control systems Dresser-Rand does a fair amount of HMI software development. Unfortunately, little time is taken to utilize Software and System Engineering methods such as requirements engineering, because of the workload and short time to completion in many projects. The consequence is often high cost efforts and time spent late in the project. These could have been reduced significantly by investing some time in requirements engineering early on.

The DR HMI applications are usually developed without any kind of requirements specifications captured in a requirements document such as an URD [User Requirements Document] or SRD [System Requirements Document] as we would find for other parts of the control system. The focus of both customer and DR lies very much on the hardware design and the logic controller software. The focus on these specific parts, that are the core parts of the system, might explain the lack of attention for the HMI requirements. New HMI applications are usually developed with the last project "setting the standard" through copy/paste/modify and a set of loosely written requirements from customer without specifics or quantifications. These are often paramount requirements covering needs well known throughout the market that are already fulfilled by the system. The result is that applications are primarily based on the views and experience of the individual developer as well as knowledge transfer from the development of preceding projects. The natural knowledge transfer from project to project is valuable, but addressing the needs of the customer and stakeholders deserves more attention.

An example is the service engineer, a stakeholder frequently using the system during field service assignments either concerning the HMI itself or some other part of the turbine control system. Because this stakeholder has not been thought of or been included in the HMI system development process, we are seeing that they are having a hard time servicing HMI and using the HMI as a service tool for the rest of the equipment.

Much experience is usually accumulated through the in-house test phases of the integrated system, some of which may be transferred into the development of new systems. However, this does not necessarily capture the needs and requirements of stakeholders beyond those of the development- and test team. The HMI system is rarely seen by the customer before the FAT of the complete integrated turbine package system, and then often only by persons from the project management or site managers and not the people that are actually going to operate the system. Although it seldom leads to a project failure, the result may be weeks of engineering work to address the various stakeholder needs identified only after the development and commissioning phase. These might be items such as color usage, screen information, system functionality, process screen layout, trend- and analysis functionality and security options. From Table 1 one can see the areas of the HMI system where there have been identified misalignments between the design solutions and the "real" need of the stakeholder(s). It becomes clear that the "problem" areas lack an adequate resolution of clearly defined requirements.

Table 1: Legacy DR HMI application emergent requirements and issue areas

	Requirements for Deployed Systems		Emergent Needs and Issues		
Software Module/ Feature/Function	Existing Requirements	Poorly Defined/Understood Requirements	Many	Several	Some
Structure & Layout	6	2			X
Navigation System	2	1		X	
Graphics & Symbols	5	3			X
Color Usage & Information Coding	12	6		X	
Security System	3	2		X	
Trend Module	7	4	X		
Historic Records Handling	2	2	X		
Alarm Handling	17	8		X	
System Feedback & Information	1	0			X
Debug and Maintenance System	1	1	X		

Developing and implementing changes after system commissioning to deal with these misalignments generally leads to large use of resources and high expenses. In general we know that the later requirements emerge in a product life-cycle, the larger the costs will be to address them (Barrese, 2007). On one commissioned system, nearly 400 hours was recently recorded used on servicing, development and implementation of fixes and changes on the HMI application trend module due to misunderstood and emergent requirements. The additional expenses and engineering hours on this system might have been avoided if important project stakeholders had been involved in the application development phase. Consequently Dresser-Rand has come to see that by catching and validating the requirements earlier in the development phase they might reduce the expenses related to upgrades and fixes, as well as development costs by transferring that knowledge to future similar projects.

1.5 Proposed Method

Early acceptance of sub-system and system requirement becomes a critical part of the process when the goal is to reduce the amount of changes after the development phase. If we do not know what the customer wants, or we only think we know what he wants, how can we know that what we are building is what is needed, expected or that it fits with the rest of the system? We must uncover the stakeholder needs and application drivers and from them derive requirements which we use to build the features and functions of the system. We also need to get early validation on these requirements as we go along designing the various components and sub-systems.

In “Requirements Engineering, A good practice guide” (Summerville and Sawyer, 1997) requirement elicitation is defined as “The process of discovering the requirements for a system by communicating with customers, system users and others who have a stake in the system development”. Communication with the stakeholders is often described as talking with and listening to the people involved with the system. The problem is that the stakeholders and the developers may go from the conversation with two completely different mental visualizations of what they have agreed upon. Thus the developers will think they know what the stakeholders want until they give the stakeholder what they said they wanted. The first time the customers or stakeholders see the results of the development process, they might reject the solution completely. They might also have reactions like “Yes, but, wouldn’t it be nice to…?” or “Couldn’t we also add……?” This is often referred to the “Yes, But” syndrome and stems from the users inability to experience the software as they might a physical device (Leffingwell and Widrig, 2000). In order to avoid these reactions at the end of the development phase we need to employ techniques that allow the developers to catch, understand and validate the user needs early on.

In the domain of human-machine-interfaces, however, capturing requirements is a particularly challenging task. Not only because, in this part of the industry, it is an area with generally too little attention, but also when you first begin to ask people about it you get just as many opinions as the number of persons you ask. Anyone with experience in operating or designing HMI systems will give you their opinion about the color scheme, navigation, process interaction, alarm handling and so on.

The problem can be approached by a combination of early prototyping, demonstrators and rapid development cycles. Prototypes can help illustrate design ideas and express its traits in a visual and interactive way. Presenting prototypes in a demonstrator session with various stakeholder groups provides a setting for discussion of proposed features, system requirements and how to approach these requirements.

The prototyping, demonstrators and feedback processes will introduce a significant amount of work and time consumption to the development phase. Scheduling and planning ahead should be done to ensure that the project is kept within the time constraints. In the planning of the project development it can be useful to create an overview of the modules and sub-systems to focus on and divide them up into manageable pieces. Demonstrator target groups, i.e. project stakeholders, and their interest in the various parts of the system, should be taken into account in this segregation. After project kick off, efforts should be made to initiate dialog with the target stakeholders and include them in the planning of demonstrator presentation. It is always a challenge getting time to talk and present your ideas and solutions. People are busy, and informing the stakeholders early on, especially those located far away, might be vital to being able to organize the demonstration sessions within the allotted time.

Software applications are getting larger in scope and more complex all the time. This usually means more sub-systems, functionality, code and interface points. As the complexity of the applications grows so does the list of requirements that supports them. It can be difficult to keep track of which parts of the system that needs the “extra attention” and be subjected to the prototyping process. Similar systems for previous projects should be studied and analyzed, not only as a source of requirements for the new system, but to identify where the “fuzzy” requirements might lie. That is, the parts of the system which were based on poorly defined or poorly understood requirements. Together with system acceptance criteria the most essential parts of the application can be filtered out. This can be of great help in the narrowing down the scope of the prototype development.

Existing systems are always an important source of requirements. If similar or current versions of the systems exist, these can be used as sources for early feedback before mockups and “early stage” prototypes of the new system are created. Using these systems as demonstrators early on can help point the development of the various application elements in the right direction and reduce the amount of iterations needed to come to the right solution.

As the project progresses regular demonstration session should be held presenting prototype-simulation of prospective designs. Depending on the evolutionary stage and type of demonstrator these sessions can be interactive (i.e. the audience is allowed to use and try out the system and its functions) or purely a presentation. In our demonstrations we facilitated the session in such a manner that discussion was allowed and a constructive debate could be held. A key part in deriving understanding of the true needs of the stakeholders is asking "why". Why does the stakeholder want the system to perform this function? Why does the stakeholder want the system to respond this way? Knowing the answers to these questions will affect the way the solutions are created and probably be closer to what the user wants if not fully understood. Observing the audience reaction is also an essential element to keep in mind when demonstrating the system. The most important response is often the one that is not said.

Follow-up questions in the form of evaluation or survey forms can be a useful tool to catch these unspoken reactions. By offering the stakeholders to fill out a questionnaire with "not too specific" questions related to the system elements demonstrated, they are given a chance to let the information sink in and provide useful feedback. Combined with measurements based on the *Likert Scale*[2] the evaluations can help keep track of the progress of reaching a satisfactory solution.

In order to get some sort of manageable data to be used in the further development, the results should be processed by mapping and structure the feedback as soon as possible after the session. Table 1 shows a simple template that can be used for this purpose.

Table 2: Managing demonstrator feedback

Session ID: [Session identification number**]**
Software Module/Feature/Function: Part of the application being demonstrated
Target Group: Stakeholder target groups(s)

Old Requirement	New / Updated Requirement	Change Action	Impact Rating	Priority Rating
Sub-System / Component: Describes sub-systems or components of the mart/module demonstrated				
Existing requirement ID and description if any	New or updated sub-system requirement description	Change action description and system impact description	Rating of change impact	Priority rating

Using a table such as the above, the requirements can be traced back to the correct version or instance of the application sub-system. The prototype demonstrated is broken down into the remarked sub-systems/functions/objects. Feedback acquired on the item is analyzed and the requirement(s) tied to the remarked item is adapted accordingly and/or additional requirements are created. Design changes resulting from the requirement update(s) are examined and evaluated with respect to the impact the change will have on the rest of the system. A discussion regarding the priority should be held between the design team and the project management. The change to the system is prioritized with regards to criticality of implementing the change versus the size, system impact and development time the change will introduce.

2 Findings of the HMI Design Process

Defining the Stakeholders

The requirements engineering techniques described in the proposed method section were applied in the development process of an HMI intended for an export compressor project for one of Dresser-Rand's larger and more demanding customers in the Norwegian sector. Dresser-Rand has previously delivered several HMI systems, with a basis in the current system platform, to this customer. Over the years after these systems have been deployed, there has been a need for a great deal of servicing and upgrades to the HMI applications, mostly due to little understood needs during the development and also emergent requirements. The goal for the project was that the prototyping and demonstrator

[2] The Likert Scale is an ordered, one-dimensional scale from which respondents choose one option that best aligns with their view. There are typically between four and seven options. Five is very common. A benefit of this method is that questions used are usually easy to understand and so lead to consistent answers

methods would reduce the number of punch list items after the FAT to a minimum and create a basis for a standardization of the HMI system.

We started out the project by identifying and defining the important HMI system stakeholders and from which viewpoint they had an interest in the system. This became important for us later in the process when we began preparing the demonstrators/models and had communications with the various stakeholder groups. For instance, a test engineer might have concerns and needs related to the mapping[3] features and utilities built into the HMI system, while this is of no interest to a regular site operator. Thus, there is no need to include the mapping feature part of the system in demonstrations or presentations aimed at the site operators as a stakeholder group. Identifying the stakeholder group relations to the major functions and modules of the application helped us with planning and scoping the information presented to the stakeholder. Overloading them with irrelevant data might have prevented constructive discussions, and us from obtaining vital feedback on items relevant to the specific stakeholder group.

Deriving Requirements from Existing Systems

Before any development began on an application for the new project, we did an investigation into legacy HMI applications developed on previous projects for the customer. We studied the applications to get an insight into which features that were developed and what solutions had been used. Furthermore we investigated which needs and requirements that lay behind these solutions, and also if these needs were fully understood.

To gain further insight into the needs of the stakeholders of the legacy applications, we wanted to know what the customer and users were experiencing with the HMI system today. These are users that have been familiarized with the system through active interaction over several years and as such have a lot of accumulated experience with the application. Although we already knew about some of the major issues with these systems, it was a source of feedback which had not been actively tapped. We enabled a dialog with these users, mainly operators, site- and service engineers, to get their opinion on the various application features. Surveys were created for each stakeholder group, adjusted to the specific stakeholder group and site application, where we queried the user on his opinion of major features and functionality and explanation of their assessments. Operators, site- and service engineers are not easy to meet face-to-face for an interview or demonstration as they are often located at site offshore or abroad. The surveys proved to be an efficient method for requirement elicitation from these groups. Through the surveys we caught several smaller issues with the current systems as well as some new needs not previously uncovered.

Legacy applications were also prepared for demonstration to stakeholders of the current project. As it turned out, the feedback from the legacy application stakeholders and the current project stakeholders were quite different regarding the legacy applications. The users of these systems were as mentioned quite familiar with their working and their remarks were directed mostly towards the functional aspects of the applications. The current project stakeholders, on the other hand, had more comments about the design characteristics of the applications, like the layout, navigation, and color usage and so on.

Early Feedback from Project Stakeholders

As opposed to previous projects, we made an effort to involve the project stakeholders early on. We did this through demonstrations of the previously mentioned legacy applications and very simple mockups and models of the application for the current project. The legacy application demonstrators were prepared for the audience along with surveys similar to those given to the users of these applications. This gave the stakeholders the chance to fill out any additional points or remarks they might think of after the demonstrator session had ended. The prototypes used in the first demonstrations were very simple mockups and illustrations of aspects like the overall layout, menu structure, general function buttons, etc. They did not contain any dynamics of interactive functionality, but they opened for very constructive discussions about specific items that were of concern to the stakeholders and allowed them to express their true needs in that regard.

[3] Mapping means tweaking the gas turbine in the various burner modes to achieve an optimal operation.

For instance a requirement was derived from a discussion with a turbine package test engineer during an early demonstration of the menu system. He would sometimes forget which turbine was shown at the process screen at the moment.[4] He therefore wanted some indication of which turbine he was looking at in addition to the page headline. This derived into a requirement to highlight the menu tab for the applicable turbine allowing the user to quickly identify to which turbine the active screen belonged to.

Data collected through the initial demonstrators helped us build a preliminary requirements document and identify key requirements. The feedback and elicited requirements derived from the early demonstrations were taken back to the development and used to drive the creation of the first structure and sub-system prototypes. Many of these sub-systems were based on previous HMI systems; however, we now had much more concrete and understood requirement specifications to facilitate the creation process.

As the project progressed, more complex prototypes were developed for the various system modules. The development process vas supported by feedback from both informal and formal demonstrators. The informal demonstrators were exercised almost on a daily basis with various representatives from the project stakeholder groups varying from one to three persons. The formal demonstrators were arranged every other week on average. They were organized for in-house stakeholders and external stakeholders (customers and users), and had a duration from one to two hours. They were attended by an audience varying from 6 to 15 persons. Our experience was that sessions with less than ten participants had more useful discussions and gave us more valuable feedback. With more attendees there were simply too many "voices" in the same room.

Informal Demonstrations

Arranging presentation and demonstrations for a larger audience, which might not even be in the same part of the country as the developer or engineer, is a challenging task. This was not entirely possible on a regular basis throughout the development. Although we wanted a validation of certain parts before proceeding with the development of a specific sub-system or component, we could not wait until we had the next chance to organize a larger demonstration for the target stakeholders. Supplementing the larger, formal sessions, we made use of informal demonstrators to "fill the gaps" and continue with the development process. We used short development cycles, made an effort to keep continuous contact with the various stakeholders and had many informal demonstrations to get early validation on minor creation steps and changes. The informal demonstrations were extremely helpful in getting feedback on application components and functions driven by the requirement specifications derived from the formal demonstrations. It could be as simple as walking over to the service engineering department with a laptop, allowing a service engineer to try out a new debugging functionality, or sending an email to the customer showing the design and color of a process symbol in various states.

In a session with the customer and system users, several participants complained about the lack of feedback and indication from command push buttons. After coming back from the formal demonstration, we started developing some concepts. Before implementing these throughout the whole application we needed to get an indication if we were going in the right direction, but we could not wait until the next session with these stakeholders. So we made some simple (dynamic) prototypes and illustrations (Figure 2) showing the button in various conditions and imbedded them in emails to the various parties. We got approval with some request for minor changes, and the development could continue.

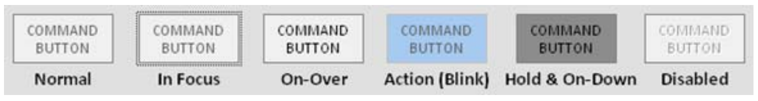

Figure 2: Informal demonstrator prototype (illustrating push-button condition dynamics)

[4] One HMI system usually includes the process screens for multiple identical turbine applications (e.g. four turbine driven generator packages).

Formal Demonstrators

The formal demonstrators were arranged by first sending out invitations to the stakeholder group representatives that were the target of the current prototype(s) and/or models. Usually these representatives had already been notified that they were going to be summoned to these demonstrator sessions to provide feedback on specific parts of the system.

Sessions typically started by an introduction of the objectives and drivers of the part(s) of the system in question, previous solution (if any) and the current approach to the problem(s). Key points for discussion were communicated to the audience, but we let discussion go during the demonstration to catch important subjects and items not recognized. In initial session where legacy applications and early system prototypes were demonstrated, we also used probing questions and tried initiating discussions on the stakeholders experience with their current system. Feedback from these discussions had a great part in identifying and defining the preliminary requirements which created a basis for the further development. The session helped us get answers to questions like:

- What are the inadequacies in the old HMI systems that drive the need for a new system?
- What parts of the old HMI system is satisfactory and should be preserved in the new system?
- What must be improved or added in the new HMI system?

The major part of the session time was spent reviewing the prototypes features and functionality, and discussing the prospective design. Scenario-based approaches were used in the demonstration, where applicable. Demonstrating typical use-case scenarios gave the audience a more realistic view of the system in real life and triggered more constructive responses. As the prototypes evolved they included more interactive features giving the stakeholders a more credible experience of the look and feel of the application. In some sessions the audience were given the opportunity to try out certain features for themselves. Human interaction is, as the name implies, the focal point of a human-machine-interface system. Actual interaction with a system gives the user a whole other perspective than only watching a presentation of its use. Allowing the users to "play" with the prototypes presented reactions that uncovered needs that would perhaps not have been perceived before the system was put in commission. An example is a function we implemented after a demonstrator session where a participant who was looking at a tag summary page, started double clicking at a row in the process tag list. He was expecting to get a popup showing him additional information about the specific tag because he was used to this from other systems. We started discussing this in the session and agreed it would a useful feature for the operators when searching for instrument tags, tracing tags back to the controller application and so forth.

If requirements are collected from a single viewpoint, they are unlikely to meet other stakeholders' requirements. We made an effort to bring different stakeholder groups together on the demonstrations when these stakeholders had an interest in the same part(s) of the system being demonstrated. Especially those who might have conflicting requirements related to these parts. Different stakeholder groups had different interest in the system in general and hence looked at different aspects of the systems during the demonstrations. This provided us with feedback from various viewpoints on the different parts. The customer project managers for instance had many observations regarding the user friendliness and availability of the system. The following comment came from one of the managers. He had experience from multiple projects and had seen how various systems were used by operators:

"Regular users with limited knowhow of the system and its more "hidden" possibilities and workings often give up very easily when trying to accomplish a task or get process information from the system. Thus this resource is lost to the goal trying to be accomplished."

Being intimately familiar with the system, developers know everything about the applications and its possibilities and limitations. Thus, what may be a simple task for us may be cumbersome or completely concealed to a regular user. For us as developers this was an extremely useful observation which drove many requirements concerning user friendliness during the development resulting in implementation like runtime error logs, info boxes, and dialogs guiding users through certain tasks.

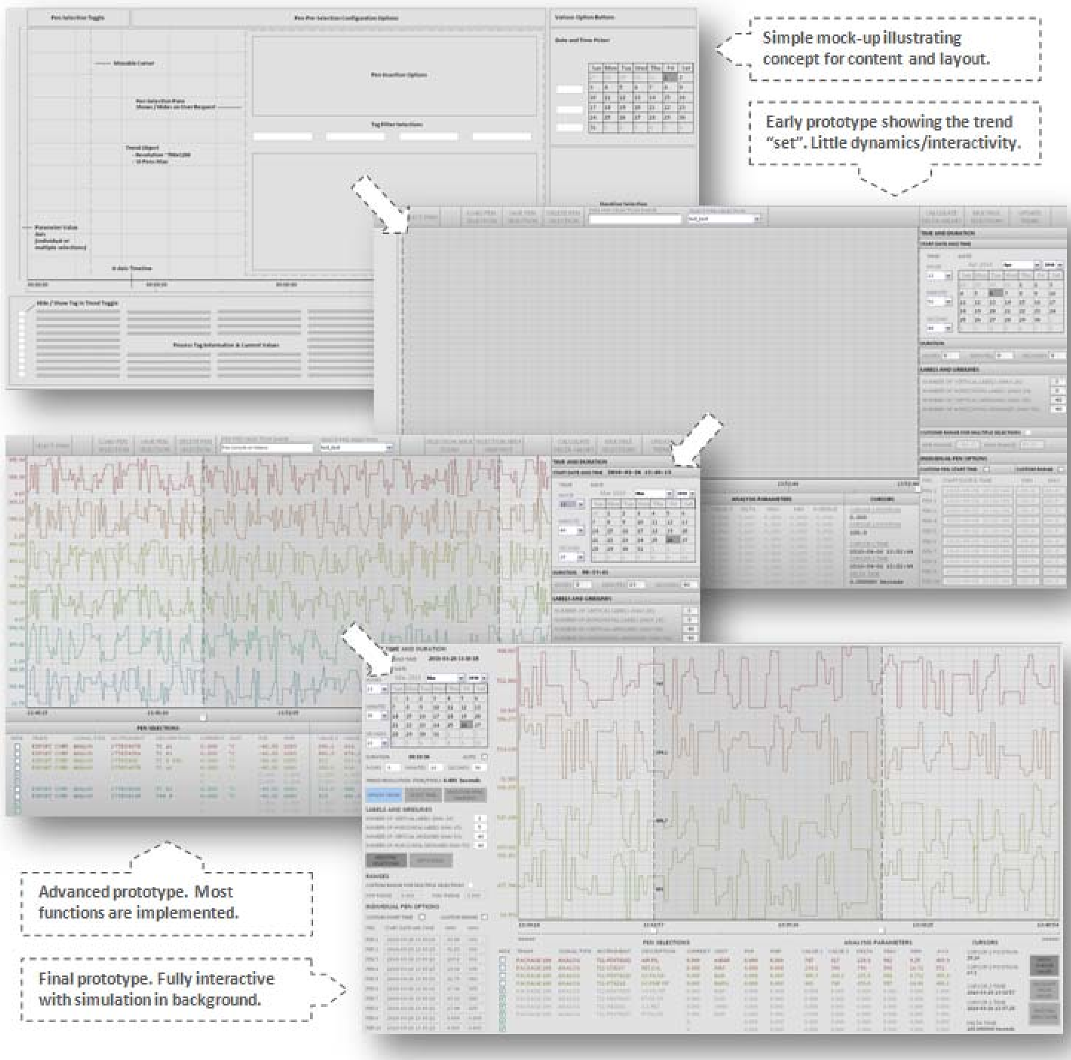
Simple mock-up illustrating concept for content and layout.
Early prototype showing the trend "set". Little dynamics/interactivity.
Advanced prototype. Most functions are implemented.
Final prototype. Fully interactive with simulation in background.

Figure 3: Trend module prototypes at various stages in the development

Observing the Audience and Collecting Feedback

The major goal of the demonstrators was collecting feedback and validation from the stakeholder audience. Feedback was not only collected through direct comments from- and discussions with the stakeholders, but also through observing their reactions on presented solutions and design choices, raising probe questions and getting explanations and clarifications on remarks. It is well known that a person's body language can tell a lot about their reactions even though they don't speak them out loud. This is something we frequently observed through our demonstration sessions. Not everyone is comfortable speaking up unless confronted directly, and it became important for us to look for these silent reactions to draw out the opinions accompanying them. For instance we made some changes in contrasts in specific pages after a discussion with a participant who we observed was peering hard trying to make out the information presented in an event list.

The feedback received during the sessions needed to be captured for further processing and configuration management. Several techniques were used including taking notes, usually done by a person other than the one leading the demonstration, writing down observations and audience reactions, surveys filled out by the stakeholders and sometimes, session recordings when a person to take notes was not available. After the sessions had ended, the feedback was typed out and sorted according to topic and sub-systems, e.g. trend point selection or valve status feedback.

Deriving and Updating Requirements by Demonstrator Feedback

Figure 3 depicts prototypes of the HMI trend module at various stages in the development. The initial prototype is simply a drawing used as a basis for discussion regarding the layout of the page, placement of objects/text/buttons, use of popups for specific functions and so forth. From the session where this mock-up was presented we obtained feedback like to limit the use of popups, include more than one cursor so values could be compared and allow for the user to specify number of gridlines.

Bringing back and analyzing the information we acquired from the mock-up we created a more evolved prototype, but still without any dynamics. The prototype was more like a façade. Easy to create, but it allowed for an exploration of the direction in which we wanted to design the module. We found it prudent to create and evaluate these simple prototypes so that we could discover usability and design problems early on before investing a lot of time and resources in developing the full functionality and final design.

In the third prototype most of the design and functions had been agreed upon. It included more interactive features and showed some functionality for the participants to try out. Some aspects were still smoke and mirrors, but it included the interactions we wanted to explore further. Derived from the demonstration of this prototype were requirements like showing graph values on the cursor, using non-gradient backgrounds and graying out information on hidden pens. By catching the tougher design and technical requirements early on with the simple prototypes, we were left with only these relatively straightforward changes at the end of the development leading to the final and validated trend module prototype.

Each demonstration left us with a lot of data that needed to be managed and derived into requirements and design specifications. Notes and recordings from the demonstration were reviewed as soon as possible after the demonstration. With the demonstration fresh in memory, we could rewrite and organize the results supplemented with our own observations and findings from the session. To put the feedback to practical use the data had to be analyzed and used to either update existing requirements or derive new requirements. Table 3 is filled out based on a sample of analyzed data from a demonstration surrounding the HMI trending functionality.

Table 3: Requirements update table for trend module

Session ID: **[**CU-006**]**
SW Module: DR-HMI Trend Module (System->Common->Historic Trend, subRec_Trend)
Target Group: Statoil SAS Team

Old Requirement	New / Updated Requirement	Change Action	Impact Rating	Priority Rating
Time and Duration Selection				
New Requirement	The user shall have the option of choosing an automatic duration calculation as the time between the chosen start date/time and the current date/time.	Update trend time selection script. Create objects and graphics for auto duration selection and script/graphics link tags.	Minor	2
RT.02.05: The user shall have an option of increasing the start time with one period.	The user shall be able to jump back an fourth in the trend object with 1, ½ or ¾ of the specified trend duration.	Update quick-jump function with duration fraction calculations. Add new graphic connections. Add quick-jump buttons to trend object.	Minor	4
Trend Object Cursors				
New Requirement	Cursor values for each pen shall be shown at the pen/cursor crossing in the trend object at user request.	Modify cursor objects in trend module. Create function(s) for cursor value placement. Create selection logic and graphics on trend screen.	Moderate	2

When managing the requirements in this way it gave us a good overview of the changes they would require in the application as well as linking the requirements back to the demonstrator session and the respective stakeholder groups. The newly derived requirement can be seen in comparison with the existing ones, if any already exist. There is a short description of the changes needing to be made as a result of the requirement update, as well as the affects these changes will have on the system. The impact rating indicates whether the change will introduce a minor, moderate or major impact on the system which gives a suggestion as to the time consumption and resources it will require implementing the change. Together with the priority rating, reflecting the stakeholder(s) need for this function/feature in the system, the table provides a good foundation for a management decision to execute the changes to the system. If a project has a tight schedule and limited resources, implementing a low priority change with a major impact on the application should probably not be given the go-ahead.

3 Method Evaluation

As it was mentioned in the beginning of this paper, little attention had been given to the HMI requirements on previous projects which might have been a contributing cause of the emerging requirements we were seeing in these systems. By initiating the process of early prototyping and demonstrations there became a whole new focus on the HMI. The demonstrators were very well received by the people involved and both customers and in-house stakeholders were pleased with the initiative. Identifying stakeholders and discussing the system with them, made people feel like they were part of the requirements elicitation process and as such they became engaged in its development. The outcome of the process was in fact that the new application was accepted and included in a frame contract with the project customer, incorporating the application in future project and upgrades.

The iterative method combination of rapid prototyping and demonstrations as well as stakeholder surveys proved very effective in eliciting and validating requirements and system design. By beginning prototyping early on and including the fuzzy requirements we managed to catch the real needs of the stakeholders early on in the development process. Thus, only minor changes were necessary towards the end of the project. As we can see from the numbers in Table 4 a lot of new requirements were derived from the described process, which affected the output system significantly from the ones previously delivered.

Table 4: Requirements and design changes from formal demonstrator sessions [5]

	Requirements		Design Changes		
Target Groups	**New Requirements**	**Updated Requirements**	**Major**	**Moderate**	**Minor**
Session ID: [UO-001]					
Users & Operators	25	10	5	13	14
Session ID: [UOCU-002]					
Users & Operators Customer Management	12	6	0	3	12
Session ID: [DE-003]					
EMEA Control Engineers (Developers)	8	7	4	1	3
Session ID: [DEDMPM-004]					
EMEA Control Engineers EMEA Management (Department) Project Management	6	13	2	4	12
Session ID: [CU-005]					
DR Service Engineers EMEA Control Engineers	8	3	0	5	3
Session ID: [CE-006]					
Statoil SAS Team (Customer)	17	11	2	10	16
Session ID: [CE-007]					
EMEA Control Engineers DR Service Engineers Project Management	4	7	0	3	5
Session ID: [CE-008]					
Statoil SAS Team	3	5	0	1	4

The above methods could be applied in any user interface development project, and especially in those starting out with a lot of vague or little understood requirements. Even though demonstrators and reviews can add considerable time to the UI development effort, time saved by catching requirements and identifying issues early is small compared to time required to fixing problems later. Involvement of important system stakeholders through the design and development teams can greatly affect the

[5] In the table Design Changes means changes to the system in relation to the prototype presented for the specific session as a result of the feedback on that prototype.

outcome of the development process. The result is more likely to be an interface that successfully meets both user needs and system requirements. Prototypes and demonstrators are as shown useful methods in this regard, but there are associated risks.

Future Research

When stakeholders are seeing that attention is given to their opinion of the system, they are often quick to put forward their “wish list”. One user’s desires might be in conflict with customer requirements and the views of other system stakeholders. Developers might also be confronted with situations where the customer or other stakeholders without necessary understanding of technological constraints or options, have a large number of requirements that are not realistic from a technological point of view. These might also be in conflict with the time-to-delivery, in which case it is not feasible to implement them. Because of organizational roles and the relationship between the development member(s) and demonstrator participants communications might go in the wrong direction and promises made that cannot be upheld. How to handle or avoid these situations should be a subject for further study.

Another issue is the conflicts that can appear by considering the concerns of multiple stakeholders. By bringing multiple stakeholder groups together we can obtain the different viewpoints of these groups and people. Although this is important when performing requirements engineering it can result in conflicting situations and requirements. Methods for how to cope with these situations and manage conflicting requirements, where negotiations and system tradeoffs is required, should also be a subject for further investigation.

4 Conclusion

Requirements engineering is a difficult and important part of any product development. In this paper it is shown how rapid prototyping and demonstrator sessions can be used as a method for eliciting and validating requirements early on in the project development phase. In the requirement elicitation process the prototypes can be used not only as a means of expressing the requirements, but also for validating, giving an understanding of the data acquired. Even simple mockups and models of the system features and functionality can be used as a means to help understanding the requirements and detect misinterpretations and incompleteness.

In most companies, communication with the stakeholders is an organizational issue which often results in a weak link between the stakeholder needs and the realization of the product carried out by the software engineers. Especially in user interface products, where consideration of the human element is essential in developing the system design, communication with and understanding the end user is vital. By applying the methods proposed in this paper the developers hve the opportunity of talking with the customers, users, and others involved with or affected by the system and understanding the rationale for their requirements.

5 Bibliography

Bahill, A. T., & Henderson, S. J. (2004). Requirements Development, Verification, and Validation Exhibited in Famous Failures. *Systems Engineering , 8* (1), 14.

Davies, P. (2004). Ten Questions to Ask Before Opening the Requirement Document. *Managing Complexity and Change* , 13.

Leffingwell, D., & Widrig, D. (2003). *Managing Software Requirements: A Use Case Approach.* Pearson Education.

Muller, G. (2008, February 21). From the soft and fuzzy to SMART engineering. *Gaudí Project* , 14.

Muller, G. (2008, February 21). What is a Good Requirement Specification? *Gaudí Project* , 11.

Paetsch, F., Eberlein, A., & Maurer, F. (2003). Requirements Engineering and Agile Software Development. *IEEE International Workshops on Enabling Technologies* , 6.

Sommerville, I., & Sawyer, P. (1997). *Requirements Engineering, A Good Practice Guide.* John Wiley and Sons.

Sutcliffe, A. (2002). *User-Centered Requirements Engineering.* Manchester, UK: Springer-Verlag.

Szekely, P. (1995). User Interface Prototyping: Tools and Techniques. 15.

Verma, D., & Penotti, M. (2005). Fundamentals of Systems Engineering (SDOE 625). *Systems Operational Effectiveness and Life Cycle Analysis* . Hoboken: Stevens Institute of Technology.

Wiegers, K. E. (2001). Software Requirements: 10 Traps to Avoid. Process Impact.

6 Biography

Jan Magnus Røkke received his BSc in Cybernetics from Buskerud University College in 2007 and in 2010 he received his MSc in Systems Engineering from Stevens Institute of Technology. He is currently employed as a Systems Controls Engineer in the EMEA Controls Department at Dresser-Rand AS.

Gerrit Muller received his Master’s degree in physics from the University of Amsterdam in 1979. He worked from 1980 until 1997 at Philips Medical Systems as a system architect, followed by two years at ASML as a manager of systems engineering, returning to Philips (Research) in 1999. Since 2003 he has worked as a senior research fellow at the Embedded Systems Institute in Eindhoven, focusing on developing system architecture methods and the education of new system architects, receiving his doctorate in 2004. In January 2008 he became a full professor of systems engineering at Buskerud University College in Kongsberg, Norway.

Dr. Michael Pennotti is Associate Dean for Academics and Distinguished Service Professor in the School of Systems and Enterprises at Stevens Institute of Technology. A systems engineering leader for more than thirty years, Dr. Pennotti has broad experience with both technical and organizational systems. He spent twenty years at Bell Laboratories designing, analyzing, and improving the performance of three generations of anti-submarine warfare systems for the United States Navy. In 1990, he shifted his focus to business management, and over the next ten years, served on the senior leadership teams of three different businesses as Quality Director for AT&T Business Communications Systems, Human Resources Vice President for Lucent Technologies’ Enterprise Networks Group, and VP Quality for Avaya. Since joining Stevens in 2001, Dr. Pennotti has helped develop the SDOE Program into the largest graduate program in systems engineering in the world. He is a member of the International Council on Systems Engineering and a senior member of both the IEEE and the American Society for Quality. He holds Ph.D. and MS degrees in Electrical Engineering from the Polytechnic Institute of New York, a BEE from Manhattan College, and is a graduate of the AEA/Stanford Executive Institute for Technology Executives.