High Dependability Computing Program

Evolving a Dependability Requirements Elicitation and Modeling Framework Based on Use

Paolo Donzelli[1]
Forrest Shull[2]
Sima Asgari[1]
Victor Basili[1,2]

Computer Science Department[1]
University of Maryland
College Park, Maryland 20742

Fraunhofer Center for Experimental Software Engineering[2]
College Park, 20742 MD, USA

November 2006

Technical Report CS-TR-4851
UMIACS-TR-2007-06

Evolving a Dependability Requirements Elicitation and Modeling Framework Based on Use -

Abstract

Correctly identifying and expressing dependability requirements for software systems has wide-ranging consequences for planning and conducting software development as well as for the final system success. Yet crucial difficulties exist, many stemming from the fact that definitions of "dependable" will vary not only from system to system, but will be perceived differently by different stakeholders of the same system.

UMD is a requirements engineering framework for eliciting and modeling dependability requirements that has been devised, to mitigate such difficulties.

In this report, we introduce UMD and describe an empirical study designed to shed some light on the feasibility of the ideas behind UMD and to identify which aspects of the framework could be improved, in the perspective that software technology transfer from research to industrial use should proceed iteratively and empirically. Subjects in the study consisted of 7 students in a graduate-level class.

Empirical qualitative and quantitative results show that the UMD approach is feasible but also allowed us to identify important missing aspects, confirming our assumption that it was not yet mature enough for a rigorous industrial study. The contributions of this study have been twofold: Demonstrating the usefulness of the tech transfer approach which we have followed as well as the feasibility of the UMD approach.

Sommario

1. Introduction

Previous papers [18] have emphasized that transferring software development practices from research to industrial use should proceed iteratively and empirically. That is, mature practices that have been initially evolved through application to testbeds should next be applied in practice, in a context that to some degree approximates the expected environment of use; the results of this application should be rigorously observed; and this observation should be used to better tailor the practice and reduce risks inherent in its adoption. Once tailored, the new version of the practice should be again observed in use in order to test whether the expected improvement is in fact observed.

In this report, we present an instantiation of the above paradigm for a practice for eliciting and modeling dependability requirements called UMD (i.e., the Unified Model of Dependability) [3,8]. This is a challenging area because:

- Requirements engineering, the branch of software engineering concerned with methods, techniques, and tools for eliciting, modeling, and analyzing software requirements, is one of the most critical areas of software development: Not only are requirements errors the most costly and time-consuming to correct, but erroneous or omitted requirements are often indicated as the main reasons for project failures [1,7,16].
- Among requirements, dependability requirements are particularly difficult to deal with for both stakeholders and analysts, as they cover many different aspects of a system at the same time [10,13,14,20], for example: failures modes and acceptable failure rates, potential hazards, recovery time and system reaction to specific failures, external events that could damage or prevent the system from functioning correctly.
- Dependability requirements are deeply rooted in the specific context [6,19]. The International Federation for Information Processing (IFIP) [9] defines dependability as "the trustworthiness of a computing system that allows reliance to be justifiably placed on the services it delivers." However, "reliance" is contextually subjective and depends on the particular stakeholders' needs. Different stakeholders will focus on different system attributes—such as availability, catastrophic failure avoidance, and deliberate-intrusion prevention—as well as require different levels of adherence to such attributes. The same attribute can also mean different things to different people, and multiple definitions of the same attribute are common [5,11,15,19].
- Precise definitions of acceptable levels of dependability are widely varying, as application is being made in unorthodox application areas. A traditional issue for space, aeronautical, nuclear plants, and defense systems, where a complete absence of failure is required, dependability is now increasingly important for systems in many other sectors, including the health care and automotive industries, as well as mainstream systems, ranging from electronic commerce to mass-marketed products. In many of these new contexts, cost-effective services are required with reasonably low failures rates, rather than a complete absence of failures, [3,4,12,17].

For all of these reasons, selecting and adopting a technology for elicitation and modeling of dependability requirements is a complex decision-making process that must be based upon firm empirical evidence showing pros and cons for the target environment. Addressing this problem thus requires an evaluation approach that allows researchers to proceed incrementally, gaining confidence in the feasibility of the technology under study before tailoring it for, and eventually transferring to, their specific industrial context. This report describes an initial study in this area that applies such an iterative, empirical approach to technology evolution. The report is organized as follows: Section 2 briefly describes UMD, highlighting its main characteristics.

Section 3 describes a study aimed at testing the feasibility of this technology, while Section 4 reports the results, and shows how these results are being incorporated into an evolved version. Finally, conclusions are given in Section 5.

2. Practice under study: UMD

UMD is a requirements engineering framework for eliciting and modeling dependability requirements. UMD is based on a modeling language that adopts a small set of basic dependability concepts (e.g., failure and reaction) to facilitate involvement from stakeholders. An active involvement of all stakeholders (i.e. those who affect or are affected by the system) is crucial for the success of the requirements engineering process [16]: Requirements are the result of a cooperative effort, in which all participants, stakeholders and analysts alike, must contribute through continuous interaction. The availability of a modeling language that stakeholders can easily grasp and adopt to express themselves and interact is thus key for the success of the whole requirements engineering process.

From this perspective, UMD aims at putting the stakeholder at the center of the requirements engineering process, providing a simple (but rigorous) language that could benefit stakeholders and analysts during elicitation and negotiation. During elicitation, stakeholders can precisely formulate their needs, while analysts can identify and highlight potential areas of improvement. During negotiation, stakeholders can better understand each other's needs and become more willing to negotiate their initial positions, while analysts can more easily identify discrepancies and suggest reconciliation solutions.

2.1. UMD modeling language

UMD is failure-centered as it permits stakeholders to express their requirements by specifying what they see as potential *failures*, or classes of failures, that may arise but which should not affect the system or specific services (i.e., the failure *scope*). Stakeholders may also quantify what they assume could be the tolerable manifestation of a failure (i.e., the failure *measure*) and the desired corresponding system *reaction*. Whenever necessary, stakeholders may also specify external *events* that could harm the system. As illustrated in Figure 1, failure, scope, measure, reaction, and event are the basic modeling concepts of UMD that stakeholders use to express their dependability requirements. For example, for an on-line bookstore system, a dependability requirement expressed using UMD could be: "The service *book search* (scope) should not have a response time greater than 10 seconds (failure) more often than in 1% of the cases (measure); if the failure occurs, the system should warn the user and recover full service in one hour (reaction)."

UMD is stakeholder-oriented, as failure, scope, measure, reaction, and event are basic concepts that stakeholders can easily grasp and associate with entities proper to their application domain. Rather than dealing with abstract entities (dependability and its attributes), stakeholders can focus on practical concepts and more effectively map their dependability needs to their context. Moreover, to better support stakeholders in formulating their requirements, and addressing the needs of a particular application context, UMD can be customized to refine the way in which these basic aspects are modeled. For example, to help stakeholders identify failures that should not affect the system or a service, we may suggest examples of the different types of failures that could occur. (Some examples are in Figure 1: response time, accuracy, etc.) Specifically, stakeholders can use the taxonomy items already available or tailor, expand, and modify them according to their specific needs.

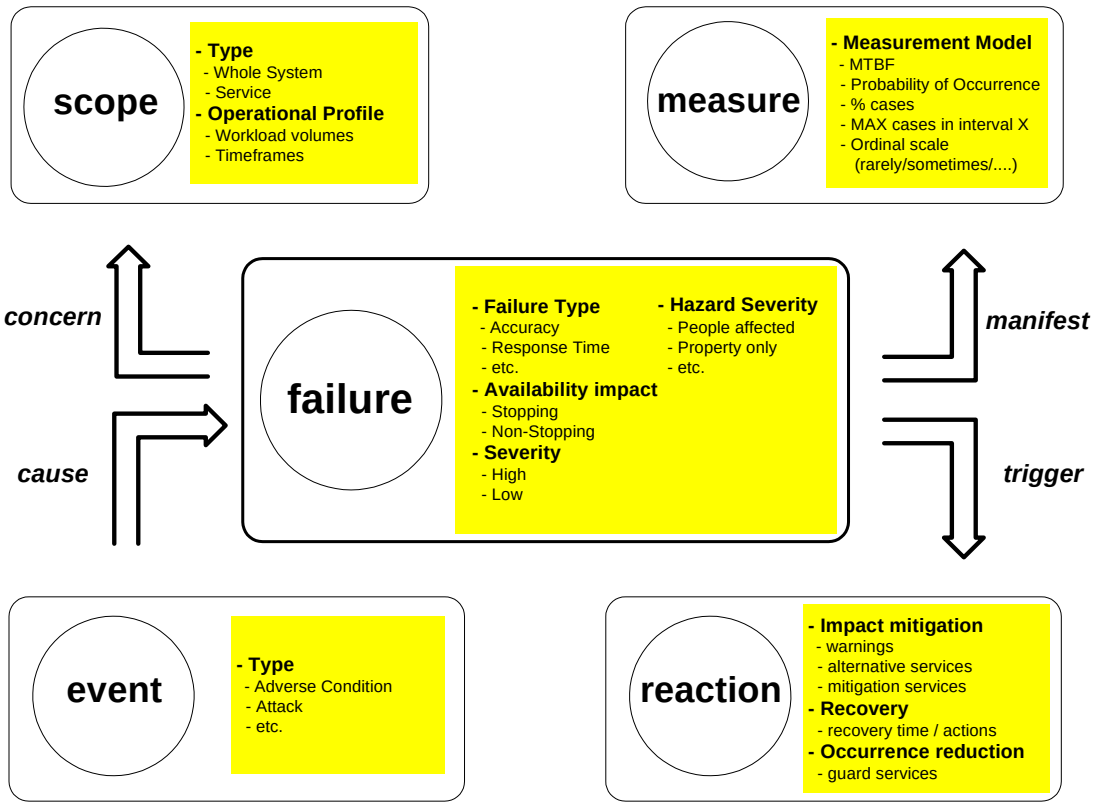

Figure 1. UMD concepts and relationships.

2.2. UMD application process

As described in more detail in other papers [2,8], UMD can be applied in different ways according to specific project needs, from helping stakeholders understand the context of and how to improve an existing system, to discovering requirements for a future system on the basis of an initial set of functional requirements. However, it is possible to identify some main activities that characterize its application:

- Initial **Scope definition**. This is the starting point of the UMD application process, aimed at defining the services of the system on which stakeholders will focus to specify dependability requirements. To identify the initial list of services, stakeholders and analysts may work in collaboration starting from an existing system or from a set of functional requirements. It is important to note that this initial list can be enriched at any point during the application process.
- **Requirements Elicitation**. Each stakeholder, supported by the analyst and guided by the UMD structure, specifies her/his dependability requirements by defining (either for the whole system or a specific service) a set of undesired failures, their tolerable manifestations, possible triggering external events, and desired reactions
- **Requirements Negotiation.** Project stakeholders often have diverse and even conflicting needs that must be reconciled in order to produce the final requirements document. In any project we can have different stakeholders, so negotiation is possible among stakeholders playing the same role or stakeholders having different roles.

2.3. UMD Tool

To implement UMD, a web-enabled tool has been developed [8], organized around two main tables for requirements collection:

The "*Scope Table"* (Figure 2), which allows stakeholders to define the services for which dependability could be of concern, starting from the system functional description.

The "*Failure Table"* (Figure 3), which allows stakeholders to specify their dependability requirements. In particular, for each service identified during the scope definition activity or for the whole system, the stakeholder can: a) Identify and describe the undesired failures (by specifying/selecting the appropriate characterization scheme, i.e., type, severity, potential hazards, etc.); b) Define the tolerable manifestations (by specifying or selecting the appropriate measurement model); c) Define or select the desired system reaction behavior (e.g., warning, mitigation, and alternative services). Finally, by using the concept of event, stakeholders may also specify (when appropriate) external events that could be harmful for the system and describe their possible impact on the system in terms of resulting failures and corresponding scope.

3. A Feasibility Study

Because the UMD framework is a new technology, we were interested in a study that would shed some light on the overall feasibility of the ideas and provide an indication of which aspects of the framework (i.e., the modeling language, the application process, and the supporting tool) could be improved.

We felt that it was important to debug these aspects at a basic level before taking it to industry for a pilot study, in which a relatively minor problem with the language or tool interface would obscure the salient aspects of the underlying concept and methods that we wanted feedback on.

Thus, the main goals for this study were to understand:

- Goal 1: To what extent the UMD framework was effective at supporting stakeholders during elicitation and negotiation of dependability requirements;
- Goal 2: How to enhance the UMD framework to improve its effectiveness.

3.1. Research Questions

The main hypothesis to investigate during this study was:

"The UMD framework (modeling language, application process, and tool) supports stakeholders in dealing with the complexity of the concept of dependability and helps them during the elicitation, formalization, and negotiation of dependability requirements".

This hypothesis led to the following specific research questions:

1. Does the UMD framework help elicitation? How and why is a stakeholder facilitated (or hindered) in expressing his/her requirements?
2. Does the UMD framework help negotiation? How and why does the UMD framework help (or hinder) the negotiation process? Can stakeholders understand each other? Are critical discrepancies among stakeholders easier to identify?

3.2. The Study

The study consisted of applying UMD to a class project with the objective of defining the dependability requirements of an online bookstore called *eBookstore*.

We used the *eBookstore* problem as we assumed this type of system would represent a familiar domain for the students, so they could act as real stakeholders focusing on their dependability needs. We did not want the results of the feasibility study to be confounded by students having to learn an unfamiliar system and possibly misunderstand the likely interests of various stakeholders. We confirmed with a questionnaire before the study began that this type of system was familiar to all students.

We considered this system to have two main classes of stakeholders:

- Customers: A customer is anyone who accesses the *eBookstore* to find and potentially buy a book;
- Sales managers: The sale manager wants customers to feel confident in using the system to search for and buy books, and to have a sufficiently pleasant experience that they will return for future business. At the same time, a manager also knows that developing a perfectly reliable system would require a prohibitive investment, so it is important to find the right balance between system cost and customer happiness.

3.3. Subjects

There were 7 subjects in this study. All were students in a graduate level software engineering class at the department of Computer Science, University of Maryland, titled: “A Quantitative Approach to Software Management and Engineering.”

The students were assigned to represent 4 Customers and 3 Sales Managers to define the dependability requirements of the eBookstore.

The level of subjects’ experience was assessed using a background questionnaire distributed at the beginning of the course. The subjects with the least past experience in software and requirements engineering were given the role of customers, while the subjects with more experience were selected to act as Sales Managers.

Subjects were told that they would be graded on the outcome of each phase of the assignment. To encourage accurate data collection and representative results, they were told ahead of time that they would not be graded in any way on the effort they expended or the number of requirements generated. Instead, the announced grading criteria were based on the completeness of their submission (meaning whether as many fields as possible were filled out for each requirement) and whether each requirement generated was supported by a well-reasoned rationale for its inclusion.

3.4. Study Design

Before running the study, subjects received a short training session, which consisted of a 2 hour-lecture on the following topics:

- Requirements engineering;
- Dependability requirements;
- Introduction to the UMD framework and supporting tool through a very simple case study.

After the training, the subjects were divided into the two groups and assigned a specific stakeholder role to represent (customer or sales manager).

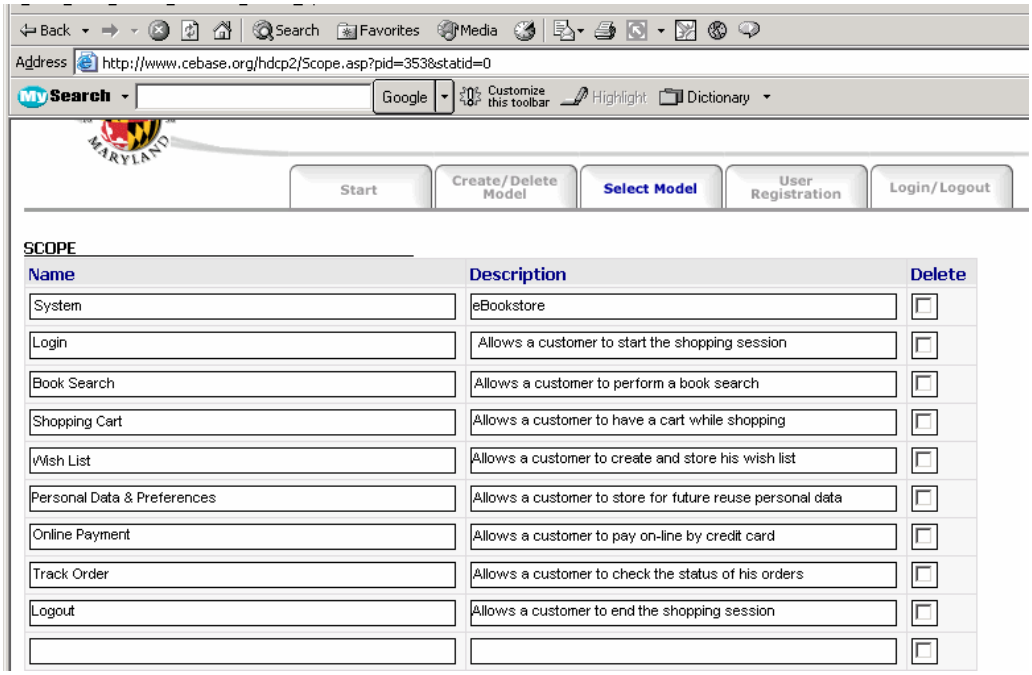

Figure 2: The UMD Tool "Scope" table for the eBookstore

Before proceeding with the project the UMD tool was initialized, by researchers, with the basic settings needed by subjects in order to provide a common environment. Specifically,

- The "scope table" tool was initialized (see Figure 2) with a set of services provided by the eBookstore (i.e., Login, Logout, Book Search, Shopping Cart, Wish List, Personal Data & Preferences, Online Payment, and Track Order). It is worth noting that these functions have been derived from the most commonly-used commercial systems.
- As discussed in Section 2, the basic taxonomies for each field of the "failure table" were customized to provide guidance to users. The taxonomies adopted for the class project are summarized in Table 1. It is important to note that students could extend or modify them at any point during the elicitation process.

Each subject was asked to apply the UMD process in a series of three treatments. Data was collected in each treatment through effort logs (see Section 3.5).

Treatment one: Each subject, guided by the structure provided by the tool, was asked to fill in as many failure tables as necessary to define her/his dependability requirements. As an example of what a table entry might look like, Figure 3 shows the failure table filled in to express the following dependability requirement, which was noted by a student using the customer role: "The service *book search* (scope) should not have a response time greater than 10 seconds (failure, characterized as non -stopping but serious) more often than in 1% of the cases (measure); if the failure occurs, the system should warn the user about the reasons, suggest a better time to try again, and recover full service in one hour as maximum (reaction)."

Taken together, these tables represent the **dependability model** of the eBookstore produced by the subject from the point of view of his/her role. Each subjects submitted their models individually, with a report explaining the reasons for their decisions, any problems using the model, and shortcomings of the model. Nine calendar days were available for completion of the assignment.

Treatment two: All subjects with the same assigned stakeholder perspective met to combine their individual models into one consensus model through negotiation. Each group submitted a unified model and a report explaining reasons for their decisions, the compromises they have made, problems using the model, and any shortcomings of the model for the unification process. One calendar week was allotted for completion of this activity. During this week two short (20-minute) discussion sessions were arranged to help subjects with problems and questions about the assignment.

Treatment three: Members of both stakeholder groups met to combine their models into one through negotiation. The class submits a unified model and a report explaining reasons for their decisions, compromises made, and other issues. The third part of the assignment took more than the one-week time frame that was originally planned. One of the researchers attended the discussion session between all subjects. During this meeting the subjects, with the researcher's help, came up with a single model for all of the stakeholders of the *eBookstore*.

Post-study follow-up: After completing the assignment, the subjects were given a questionnaire with focused questions to elicit their experience on various aspects of the process and tool (see Section 3.6, qualitative analysis).

Failure characterization: Failure Types: • Functional correctness: System or service does not work or it does not implement the functional requirements. • Throughput: Average or peak number of items (e.g., books) per unit of time dealt with by the system or service is less than expected. • Response time: Response time of the system or the service greater than expected. • Static load: Max number of items handled by the system or the service is less than expected. • Accuracy: The accuracy (e.g. type of suggested books) of the system or service is less then expected. • Data freshness: The frequency of data updating is less than expected. • Confidentiality: Unauthorized disclosure of information by the system or a service. • Integrity: Unauthorized data alterations by the system or a service. • Usability: The easiness of using the system or the service is less than expected. Failure Impact over Availability: • Stopping: Failure makes the system or service unfit for use. • Non-Stopping: Failure does not make the system or service unfit for use Failure Severity: • High severity: Failure has a major impact on the utility of the system for the operator. • Low severity: Failure has a minor impact on the utility of the system for the operator.	**Event characterization**: • Adverse Condition: Any unintentional event that could have some effect on the system. • Attack: Any intentional action carried out against the system. **Measure characterization**: Measurement Models: • Mean Time Between Failures (MTBF) • Percentage of cases **Reaction characterization**: Services Types: • Warning Services: Warn user about the situation. • Alternative Services: Provide alternative ways to perform same tasks. • Mitigation Services: Reduce issue impact on the user. • Guard Services: Reduce probability of occurrence of the issue. Recovery Behavior: • Mean Time To Recover (MTTR) – Max Time To Recover (MaxTTR)

Table 1: UMD guidelines for eBookstore

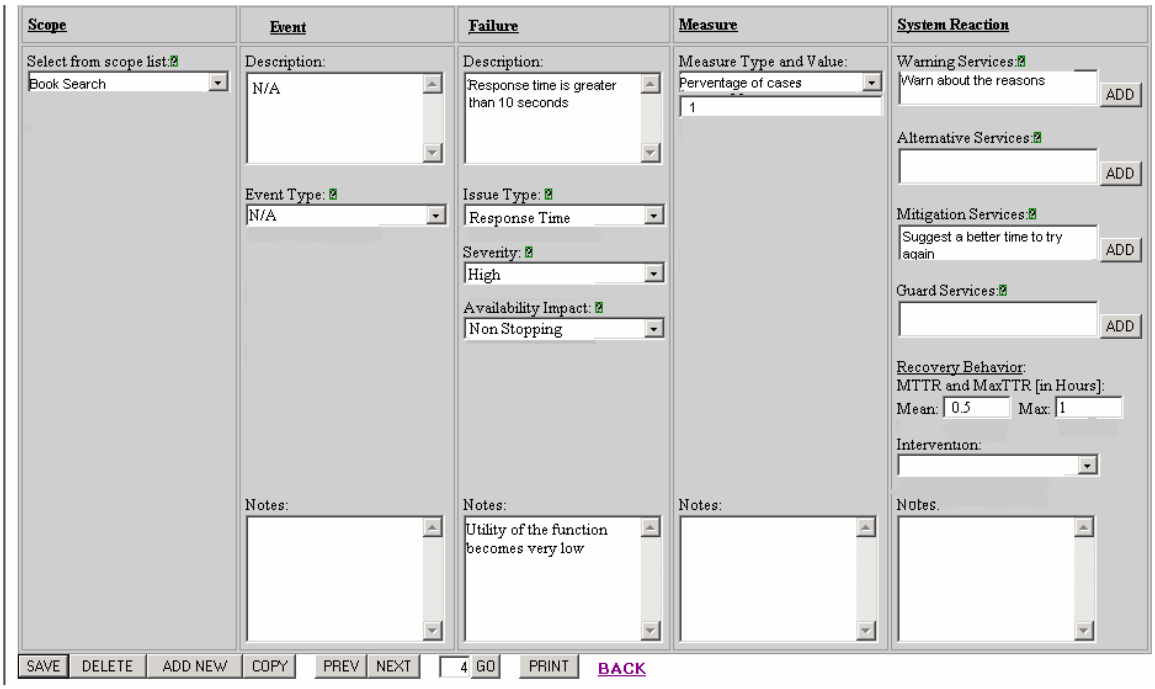

Figure 3: Example of completed "Failure" table

3.5. Data Collection

The following types of information were collected:

Information describing subjects' background. A background questionnaire was distributed prior to the start of the study to ensure that students were familiar with the eBookstore domain and to assess their level of familiarity with requirements engineering and software engineering, which was used to assign the stakeholder role as described in Section 3.2.

Individual and group effort log. The number of hours spent on each activity was self-reported by the students. We tried to maximize the accuracy of the data by announcing ahead of time that no part of the grade would be based on the effort reported.

Size of the dependability models. To measure the size of the dependability models created during the project (i.e., the models created by each student for Part 1, the models created by each group for Part 2, and the model created by the whole group for Part 3), we have adopted as a measurement unit the Line of Dependability (LOD). Each LOD represents a unique dependability requirement as captured by a failure table (see for example Figure 3).

Negotiation dynamics (how requirements were selected, improved, merged, created, and dropped during negotiations). There are no features in the UMD tool specifically aimed at supporting negotiation among stakeholders to produce a consensus set of lines of dependability. However, the formality imposed by the UMD modeling notation is aimed at facilitating the discussion by both providing a common structure to make comparisons easier, and helping identify under-specified or missing pieces of information. Moreover, the tool provides a way of organizing, manipulating, and editing the various LOD that simplifies the overall task. By explicitly storing each LOD (i.e., a completed failure table), the effects of negotiation can then be studied by comparing different versions of the dependability model. More precisely, when individuals bring their own dependability models and meet to achieve consensus, for each LOD recorded by an individual one of the following can occur, as illustrated in Figure 4:

1) The LOD from one model is accepted into the consensus model, with no changes. We call this a “Winner LOD.”
2) The LOD from one model is merged with another one from a different model, in which case it can be considered a “partial winner,” specifically:
   a) “Partial winner A,” if it appears in the consensus model with extremely minor changes (e.g., only one field of the LOD is modified by using values from another LOD, usually a Winner C)
   b) “Partial winner B,” if it appears in the consensus model with important changes (e.g., this LOD has been merged with another LOD).
   c) “Partial winner C,” if it appears in the consensus model with extremely important changes (e.g., only one field of this LOD has been maintained and has contributed to complete another LOD, usually a Winner A). .
3) The LOD is dropped; it does not appear in even modified form in future versions of the model.
4) A new LOD is created due to discussions among the team that had not been previously recorded by individuals.

A large number of winners may indicate that the individuals brought many different perspectives to bear on the final system, so that the unified set of their lines of dependability is truly necessary to represent the system fully, or it may indicate that participants in the meeting did not truly negotiate the final model and tended to accept ideas without discussion.

Finally, by comparing LODs in the different models created during the study, we can also measure whether LODs grew more complex or complete during discussion, that is, had more of their fields completed after group negotiation.

Post study questionnaire. This questionnaire captured qualitative information that was not reflected in the other metrics collected, for example, concerning what difficulties students felt that they faced when applying UMD and what kinds of strategies they used to work around them.

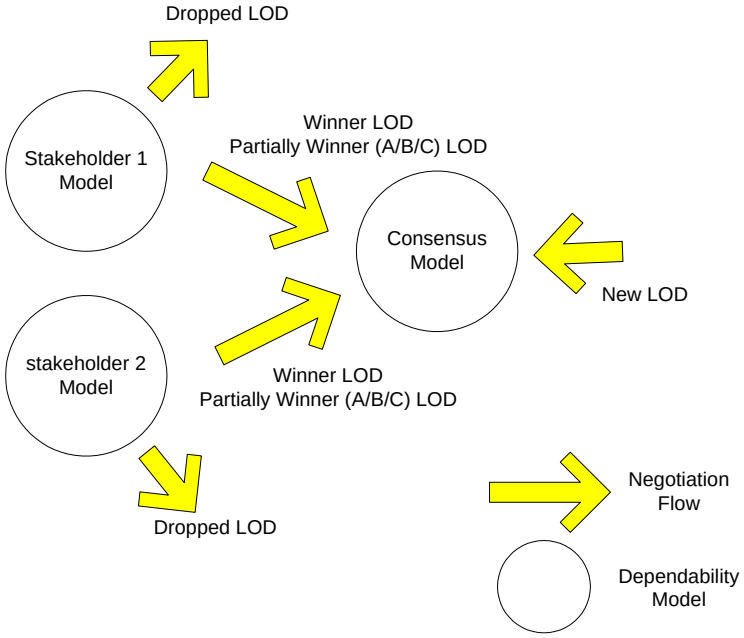

Figure 4: Negotiation Dynamics

3.6. Data Analysis

Using the measures described above allowed our analysis to use a mix of quantitative and qualitative information, which was necessary to get a better understanding of the feasibility of UMD and to address our goals of maturing the technology for use in production environments.

Quantitative Analysis.

Model size. Figure 5 describes the size of the dependability models generated at each step of the elicitation and negotiation process, using the number of LOD in each model. The 4 subjects representing customers generated individual dependability models with 22 LOD on average; their consensus model contained 33 LOD. The 3 subjects representing managers generated individual models with 23 LOD on average and 42 LOD in the consensus model. The final dependability model representing both stakeholder groups contained 62 LOD.

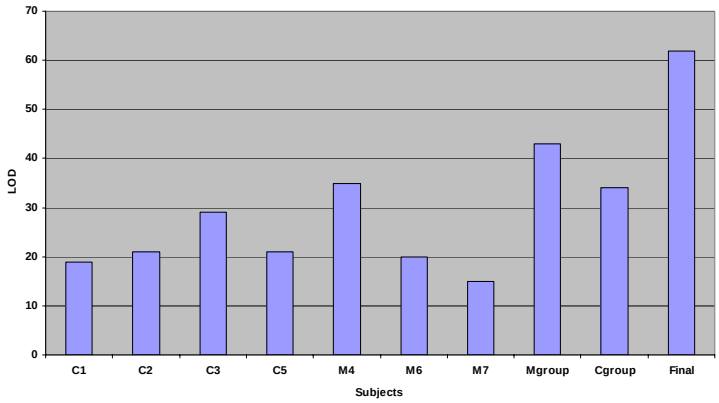

Figure 5: Size of models (in LOD)

Process Dynamics: Intra-group Negotiations. Figure 6 describes the results of the negotiations within each group, as measured using the categories defined in Section 3.5.

The 4 subjects using the customer role reported a total of 90 LOD in the first step of the process. Of these:

- Over one-third (38%) were dropped during the creation of a group consensus model;
- 7% were accepted without modification, contributing 6 LOD to the customer model (winner LODs);
- The majority (55%) were combined through negotiation and compromise: most of them (37%) as winner A, the remaining as winner B and C (12% and 6% respectively). The 49 LOD in these categories were eventually combined into 24 unique LOD in the customer dependability model.

Also, during the discussion 3 new LOD were created, bringing the total size of the consensus model to 33 LOD.

A similar pattern characterizes the negotiations among managers. 70 LOD were created by individuals. Almost a quarter (23%) were dropped entirely, while a minority (16%) were adopted as is. The remaining 61% represented 43 LOD that were combined into 32 in the final model. (No new LODs were created by the managers during discussion.)

These results show that, while subjects applying the same perspective were all focused in a similar area, there were a minority of important individual contributions that were accepted as they were originally written, without changes based on another subject's results. In both cases, a

surprisingly large number of LODs were simply dropped from consideration in the consensus model, signifying that there were cases where individuals in their own analysis took an extreme position too far outside the group consensus. These results do provide clear evidence that negotiation was in fact being undertaken, i.e. that subjects took their responsibility to create a consensus viewpoint quite seriously.

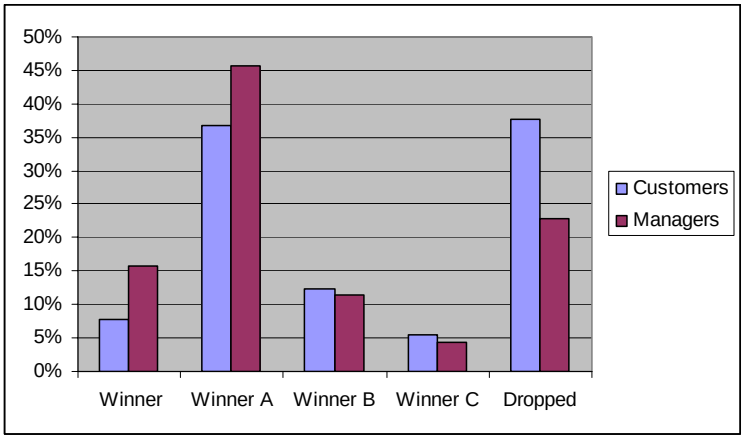

Figure 6: Intra-group Negotiation

Process Dynamics: Inter-group Negotiation. Figure 7 describes the results of the negotiations between the two groups, showing the different categories of LODs making up the final model and the contribution made by the two groups.

Recall that the customers entered negotiations with a consensus model containing 34 LODs; the manager model was roughly similar in size with 43 LODs. Based on the results, it does not appear that either stakeholder group dominated the negotiation process.

Both groups had a large percentage of their LODs carried into the new model as winners: 52% of the final model is made up of winner LODs, 44% of which were taken from the customers' consensus model and 56% from the managers' consensus model. Both groups also had a minority that were dropped completely (8% for customers and 5% for managers). The remainder of each groups' LODs were combined through negotiation: 18% of the final model's LODs are of the category winner A (33% derived from the customers' model and 67% from the managers' one), while 17% are of the category winner B (63% from the customers' model and 37% from the managers' model). This resulted in 27 different LODs transforming into 12 LODs in the final model describing eBookstore dependability requirements.

These results imply that the two groups focused on slightly different areas of the system behavior, so more than 50% of the requirements could be taken as originally formulated The final dependability model consisted of 61 LODs, among which only 3 new LODs were created during negotiation.

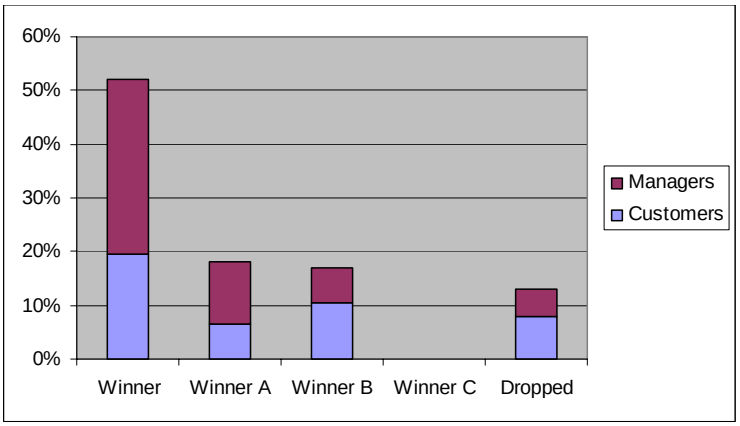

Figure 7: Inter-group Negotiation

Effort data. The effort required for each subject to generate their individual models of system dependability varied widely but were on average very similar between the two stakeholder groups: 7 person-hours for customers and 7.6 person-hours for managers.

Negotiation was an equally time-consuming task. The time required for negotiation was very similar for both groups (8 hours for customer and 8.25 for managers), although to the best of our knowledge they did not coordinate on the time required. This time investment was larger than the time that 5 out of 7 students spent on their individual model building.

As might be expected, negotiations between the two stakeholder groups were more time-consuming than the time spent within each group. We expect this is due to the need to compromise between two very different views of system functionality. It took 12.25 hours to build the final model representing both groups.

Figure 8 summarizes the overall cost of the elicitation and negotiation process by showing the effort spent on average to obtain each LOD of the customer model, of the manager model, and of the final dependability model. These measures are cumulative, that is, they include all effort spent within a phase plus that spent in all previous phases. We can see that most of the effort (about 70%) was spent in developing the models for each stakeholder group, while the negotiation to produce the final model was relatively efficient.

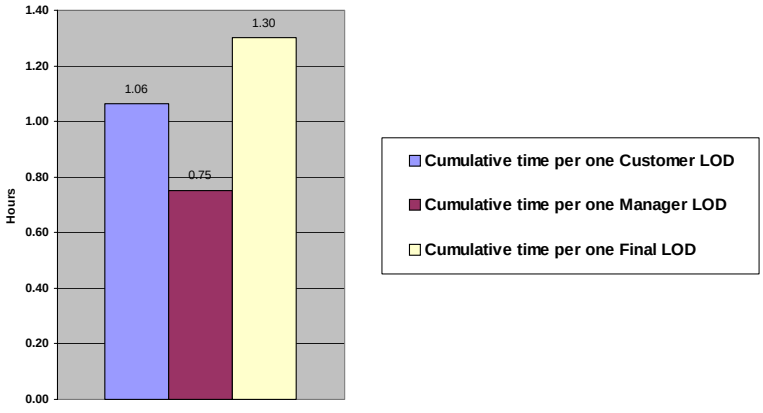

Figure 8: Average effort per LOD

LOD completeness. We define a LOD as complete when all its main fields (scope, failure, measure, and reaction) have been given a value. LOD completeness is thus an index of the quality of a set of dependability requirements, as a greater degree of completeness means that more aspects have been taken into consideration by stakeholders. We can observe that LOD completeness increased during this study since, whereas complete LODs comprised an average of 20% of the models created by individual subjects, in the final dependability model the measure was 80%.

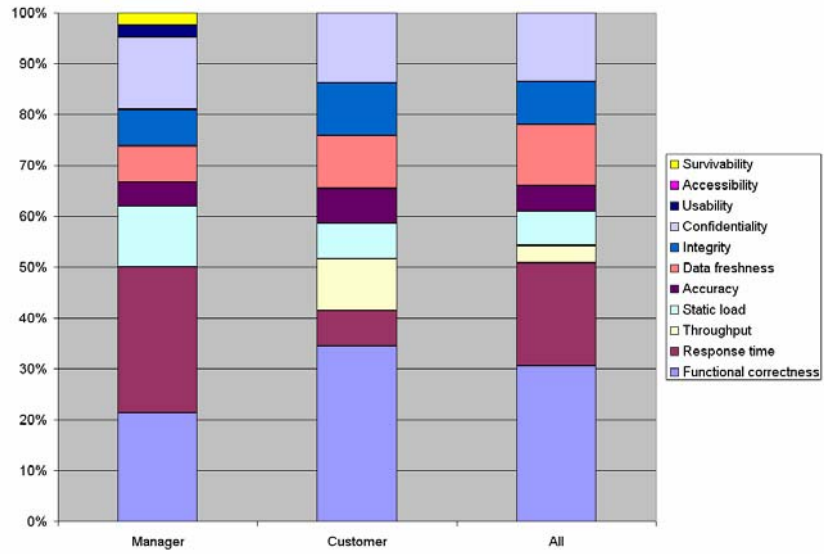

Figure 9: Failure types distribution

Qualitative Analysis.

Analysis of LOD by group. Figure 9 shows the distribution of failure types in the various consensus models (customer perspective, manager perspective, and final). While these numbers do not permit a statistically rigorous comparison, the figure illustrates that the overall breakdown of concerns was fairly comparable for both stakeholder groups. Customers did focus a larger percentage of their LODs on functional correctness and throughput while managers focused more on response time.

Analysis of subject rationales and comments. Since we were interested in gaining information that could be used to evolve the UMD modeling approach and the accompanying tool support into a version that would be better suited for use on industrial projects, we analyzed the qualitative information that students reported in their assignments and in the follow-up questionnaire. We summarize their answers focusing around some important questions:

Q 1) Was the UMD approach useful for elicitation?

Students reported a number of general points that could be improved, such as removing some ambiguity from definitions. Some deeper points touched on the nature of the approach:

- It seemed good for novices: “It helped to have a standard format in which requirements are written and to emphasize issues that I would not have known to include in the requirements

(e.g., response time).” “The UMD approach helped a lot in expressing the dependability requirements (once I have got the hang of it).”

- The UMD template (failure table) helped to make information explicit and comparable: “The model helped viewing the problem in a specific way that made various aspects of dependability described in a unified manner.” “Since everything was so structured, it was easy to break down stakeholders’ demands into their parts.” However not all subjects found the format appropriate: “I had trouble expressing functionality requirements using the given fields”
- There were some difficulties expressed by different subjects about various fields in the template. However, there was no consistent agreement on these and most of the difficulties seem to have been overcome in time.
- It helped by creating a mini-“experience base” that made requirements explicit and traceable: “UMD approach helped by organizing our thought process and codifying failure information in a way that can be analyzed later”
- The most difficult part was “identifying the best measure (we had no information on the measure and it was difficult to negotiate).”

Q 2) Was the UMD approach helpful for negotiation?

- Overall, having a common format seemed to help the negotiation: “UMD approach helped in combining requirements, since everyone’s requirements could be easily understood by everyone else”
- However, it was hard to resolve some of the ambiguities about details of the template definitions between the two groups of stakeholders: “Different individuals had different interpretations of what various fields in the model meant. This became even worse given that manager and customer groups arrived at different consensus”
- Differences between the groups were made explicit, so they could not be omitted or deferred until later. “The model helped us very well in combining requirements, because we already had models in the same format (it would have been more difficult to start from scratch and reach same completeness).” “The greatest strength of UMD is being able to merge requirements. We had no trouble identifying potential conflicts and resolving them.”
- However, resolving the issues that separated the groups was not easy: “Agreement on failure criteria [failure types and measures] was the most difficult aspect. Managers designed a very lenient system which was not satisfactory for the customers; managers did not want to design anything that would have add to their expenses; customers had to lower their expectations; few managers requirements were updated to increase customer satisfaction”
- In general, conflicts between the groups were resolved by going with the more lenient criteria – which may not always be the optimal solution for the system as a whole. “Failures were very similar, only the degree was different, so we went with the most lenient criteria”

Q 3) Suggestions for new tool requirements.

There were some low-level suggestions about how the functionality could be improved; e.g. some functions like printing simply took longer than was convenient. More sophisticated commentary on tool functionality included the following:

- It was necessary to have better connections between related information in the tool: “It was hard to look at [multiple] requirements [at the same time], navigate between requirements and scopes.” “Most difficult part was scrolling across the page to reach the different aspects (guard services, measure, etc.)”

- It was necessary to have better connections between the different models being created, to help with negotiating combined models: “Difficult to compare requirements within and between models (cannot copy requirements from one model to another).” “No way to transfer data from one model to another (we had to re-enter a good part of data by hand).” “There is not even a merge option in the tool!”

4. Lessons learned

Based on observation performed during this feasibility study, we now can provide the following information to address our study goals:

Goal 1: To understand the extent to which the UMD framework (i.e., the modeling language, the application process, and the supporting tool) was effective at supporting stakeholders during elicitation and negotiation of dependability requirements. As a feasibility study, the data collected from our subjects directly addressed this issue. The quantitative data collected showed that students were indeed engaging in model-building and negotiation during the different phases of this project. The data show that compromises were being made (i.e. previous LODs were being modified or dropped while consensus was being sought). Also, as might be expected, the data show that activities and results were different depending on whether subjects were negotiating with others using the same stakeholder perspective or a different one. Moreover, the effort required for any of these tasks was not prohibitive.

Thus the quantitative data strongly indicate that students found it relatively easy and feasible to use the modeling language to express their needs and to negotiate a common view. These indications are corroborated by the qualitative data. In their responses, multiple subjects mentioned that UMD helped with the *mechanics* of the negotiation process (i.e. the use of templates made sure that various stakeholders were identifying similar types of information, and those types were easy to compare) even if reaching an agreement between two different stakeholder groups was still difficult.

Given the status of the tool as an early prototype, it was expected that many suggestions for enhancements would be received. The nature of the comments (specifically the ones regarding the need for more connections among various types of information) will help improve the functionality based on observation of actual use and the way in which users tried to interrelate information during negotiations.

Goal 2: To understand how to enhance the UMD framework (i.e., the modeling language, the application process, and the supporting tool) to improve its effectiveness. The study has provided us useful feedback on concrete improvements necessary to the UMD framework. In particular, the following critical areas have emerged upon which we need to focus our attention:

Supporting stakeholders during elicitation. Difficulties encountered by stakeholders could be reduced by augmenting the level of guidance, for example, by providing a reference list of suggested measures that are appropriate for defining the level of tolerable manifestation for different types of services and quality constraints. In this way, the tool could provide an evolving “experience base” in which future LODs can be defined based on measures and other component pieces that have proven useful in the past. Initial work has been undertaken to explore this possibility and validate this idea [8].

Improving support for negotiation. Based on these results, we have a better idea what activities occur during negotiation: e.g. the percentage of LODs that are likely to change as a result of negotiation, and how completeness of LODs varies over time. This information gives us a better understanding of how to enrich the tool to improve its support to stakeholders. In particular, due to the structure of the UMD modeling language, stakeholders tend to focus during elicitation on each single LOD and have difficult in dealing with many LODs at the same time.

Based on these observations, the tool has been improved and is under evaluation [8]. The new UMD tool allows for analysis based on two different views across the entire set of requirements expressed by the stakeholders:

- The **Visual Query Interface (VQI)**, developed at the Fraunhofer Center Maryland and based on the idea of the Starfield display [21], is used to analyze requirements geometrically, i.e. to analyze characteristics of sets of requirements rather than individual requirements. VQI allows the spatial visualization of the requirements distributed according the values of two or more of their attributes (e.g., failure type, availability impact and severity; hazard severity; type of external event, type of reaction). Different symbols, colors, labels, and sizes can be used to highlight the attributes of interest.

- The **"Dependability Analyzer",** a prototype tool, developed by combining features provided by *MS Excel* and *Matlab* [22], is used to visually represent the emerging system dependability properties. The measures expressing the tolerable manifestation for each of the identified issues are combined to provide "aggregate values of dependability": for example, the aggregate MTBF of all the failures, or the MTBF of all the failures that are also stopping failures.

Finally, information acquired by studying negotiation dynamics has allowed us to identify the requirements for "distance metrics" that could be used to actively support a more structured negotiation process, potentially reducing the effort for this phase of application. In particular, by applying distance metrics the tool will be able to identify clusters of similar LODs from multiple models. On the assumption that LODs within a cluster are likely to be those which could be merged in the consensus model or contain discrepancies that need to be reconciled, the negotiation process can become more efficient as analysts and stakeholders can focus on each cluster in a systematic way.

6. Conclusions

This study is a demonstration of the need to apply an iterative, empirical approach to technology evolution. The quantitative and qualitative results confirm our initial feeling that, although promising, the technology and especially the tool support underlying the UMD approach was not yet mature enough for a rigorous industrial study. In fact, our student subjects helped identify some important missing aspects, such as a need for better links among similar information within UMD models to support negotiations, which should be improved before industrial teams apply the approach. At the same time, the data collected in this study do show that the approach is feasible and worth further investigation.

In terms of future work, the directions for further research described in the previous section show that the feasibility study was indeed useful for identifying concrete areas in which more functionality and hence more experimentation were needed.

Thus, we feel that the contributions of this study have been twofold: Demonstrating the usefulness of the tech transfer approach which we have followed as well as the feasibility of the UMD approach and supporting measures.

Acknowledgements

The authors wish to acknowledge support from the NASA High Dependability Computing Project under cooperative agreement NCC-2-1298.

The authors wish to thank the researchers on the HDCP project team for their insights and suggestions, and also the students who performed as subjects in this study for their hard work.

7. References

[1] A. Finkelstein, J. Dowell, A Comedy of Errors: The London Ambulance Service Case Study, Proc. Of the 8th International Workshop on Software Specification and Design, pp. 2-5, 1996.

[2] Basili, V., Donzelli, P., Asgari, S. The Unified Model of Dependability: Putting Dependability in Context, IEEE Software, Vol.21, Issue 3, Nov/Dec 2004.

[3] Boehm B., Huang L., Jain A., Madachy R., The ROI of Software Dependability: The iDave Model, IEEE Software, Vol. 21, Issue 3, May/June, 2004.

[4] Boehm, B., Huang, L., 2003. Value-based Software Engineering: A Case Study, IEEE Computer, Vol. 36, Issue 3, March, 34-41.

[5] Boehm, B., Huang, L., Jain, A., Madachy, R., 2003. The Nature of Information System Dependability – A Stakeholder/Value Approach, Technical Report, University of Southern California, CA, US.

[6] Chung, L., Nixon, B., Yu, E., Mylopoulos, J., 2000. Non Functional Requirements in Software Engineering, Kluwer Academic Publisher.

[7] D.R. Lindstrom, Five ways to destroy a development project, IEEE SW, pp. 55-58, September 2003.

[8] Donzelli P., Basili V., A Practical Framework for Eliciting and Modeling System Dependability Requirements: Experience from the NASA High Dependability Computing Project, Journal of Systems and Software Vol 79/1 pp 107-119, 2006. DOI information: 10.1016/j.jss.2005.03.011

[9] International Federation for Information Processing Working Group 10.4, www.dependability.org.

[10] HDCP - High Dependability Computing Project, 2002. http://hdcp.org.

[11] J.C. Laprie, ed., Dependability: Basic Concepts and Terminology—Dependable Computing and Fault Tolerance, vol. 5, Springer-Verlag, 1992.

[12] Littlewood, B., Stringini, L., 2000. Software Reliability and Dependability: a Roadmap. In: Proceedings of the ACM Future of Software Engineering conference, Limerick, Ireland.

[13] Melhart B., White S., Issues in defining, analyzing, refining, and specifying system dependability requirements, IEEE Conference on the Engineering of Computer Based Systems, April 2000.

[14] Mellor, P., Failures, Faults and Changes in Dependability Measurement, 1992. Information and Software Technology, Vol. 34, Issue 10, October, 640-654.

[15] Randel B., Dependability, A unifying concepts, Proceedings of Computer Security, Dependability and Assurance: from needs to solutions, York, UK & Williamsburg, VA, USA, July & November 1998.

[16] Robertson S., Robertson J., Mastering the requirements process, ACM Press Book, Addison Wesley, 1999.

[17] Shaw M., Everyday Dependability for Everyday Needs, Supplemental Proc of 13th International Symposium on Software Reliability Engineering, Maryland, 2002.

[18] Shull F., Carver J., and Travassos G. H., "An Empirical Methodology for Introducing Software Processes", *In Proceedings of* European Software Engineering Conference, September 2001, pp. 288-296.

[19] Sommerville, I., 2003. An Integrated Approach to Dependability Requirements Engineering. In: Proceedings of the 11th Safety-Critical Systems Symposium, Bristol, UK.

[20] Virtanen S., Reliability in Product Design – Specification of Dependability Requirements, IEEE Reliability and Maintainability Symposium, 1998.

[21] Ahlberg C. Shneiderman B., Visual information seeking: Tight coupling of dynamic query filters with starfield displays, Proc. CHI'94 Conference: Human Factors in Computing Systems, ACM, New York, NY (1994), 313-321

[22] Matlab, http://www.mathworks.com