Articles

Akhil Kumar Daida

9408271238

akda15@student.bth.se

Seminar 1

Article: Gorschek & Wohlin "Requirements Abstraction Model"

Article gives an overview of the RAM model. Requirements engineering using the RAM model involves the following three basic steps. The first step (Specify) involves specifying the initial (raw) requirement and eliciting enough information about it to specify a number of attributes. The second step (Place) is centered on what abstraction level the specified requirements resides on and last (Abstraction) each requirement goes through a work-up.
Step 1: Specify
Four main attributes are specified in the initial step. The goal of this step is to get an overview of the raw requirement to the extent of it being understood by the Product Manager performing the continuous requirements engineering.
The attributes are description, Reason/benefit/rational, Restrictions/risks and title.
Step 2: Place
RAM consists of a number of abstraction levels. This step involves analyzing what level a requirement is on and placing it on this level. Four abstraction levels are Product level, Feature level, Function level and Component level. The Product level is the most abstract level. Requirements on this level are goal-like in nature, i.e., not fitting the normal definition of a requirement. The Feature level is the next level in the model. The requirements on this level are features that the product supports. The Component level is the last level of abstraction in RAM. Component level requirements are of a detailed nature depicting information that is closer to how something should be solved.
Step 3: Abstraction
This third step of RAM involves abstracting and/or breakdown of a requirement, depending on the initial placement of the original requirement. The work-up process involves creating new requirements (called work-up requirements) on adjacent abstraction levels or linking to already existing ones, depending on the situation.
As part of its development, RAM was validated in industry through both static validations, getting feedback from professionals working with requirements engineering and product development, and through dynamic validation, giving a real live test-run of the model. The validations were performed to assure that the model complied with the needs of the industry. During the validations it was ascertained that requirements did come in on different levels of abstraction, and that specification, placement and work-up were feasible in a real live requirements engineering situation.

Article: J. Karlsson, K. Ryan, “A cost-value approach for prioritizing requirements”, IEEE Software, 1997.

This paper discusses about cost-value approach which prioritizes requirements according to their relative value and cost. Based on the information, software managers can make decisions such as which requirements to be excluded from the first release to keep the time-to-market at minimum. There are five steps to prioritizing requirements using the cost–value approach.

1. Requirements engineers carefully review candidate requirements for completeness and to ensure that they are stated in an unambiguous way.
2. Customers and users (or suitable substitutes) apply AHP’s pairwise comparison method to assess the relative value of the candidate requirements.
3. Experienced software engineers use AHP’s pairwise comparison to estimate the relative cost of implementing each candidate requirement.
4. A software engineer uses AHP to calculate each candidate requirement’s relative value and implementation cost, and plots these on a cost–value diagram.
5. The stakeholders use the cost– value diagram as a conceptual map for analyzing and discussing the candidate.

The Analytic Hierarchy Process compares alternatives in a stepwise fashion and measures their contribution to your objective. Using AHP for decision making involves four steps.
Step 1. Set up the n requirements in the rows and columns of an n x n matrix.
Step 2. Perform pairwise comparisons of all the requirements according to the criterion.
Step 3. Use averaging over normalized columns to estimate the eigenvalues of the matrix.
Step 4. Assign each requirement its relative value based on the estimated eigenvalues.
The redundancy of the pairwise comparisons makes the AHP much less sensitive to judgment errors; it also lets you measure judgment errors by calculating the consistency index of the comparison matrix, and then calculating the consistency ratio. The consistency index (CI) is a first indicator of result accuracy of the pairwise comparisons.
CI= (l max- n)/(n -1)

Results of the case study from the article shows that the cost–value approach is useful for prioritizing requirements. It may also be especially applicable as an aid for requirements selection, since both the value and cost of requirements can vary by orders of magnitude. Such differences are effectively visualized through cost–value diagrams, which let management take action to maximize stakeholder satisfaction. The diagrams can also be used to prioritize requirements over several release cycles. Communications between customers, users, and project managers are always likely to be difficult. A clear understanding of the choices involved in requirements selection can greatly assist this communication.

Seminar 2

Boston matrix: Yuanping & Dengsheng “Design and Implementation of a Scientific Research Funds Analysis
Model based on Boston Matrix”

Boston matrix, also known as four-quadrant analysis or BCG matrix is an analysis method proposed in the 1960s by the Boston Consulting Group which is a large consulting firms in the United States. Boston matrix is a common analysis method for enterprise product plan or business portfolio, and it can help providing a reference when generating future business development strategy. The method proposes that the basic factors determine the product mix are generally including market attraction and business strength, so it selects sales growth rate and market share of the two indicators to examine. Different combinations of the two indicators can be constructed in four product types: type question (Question), namely higher sales growth, low market share of products; star-type (Star), namely higher sales growth, compared with high market share of products; Taurus type (Cash Cow), namely a lower sales growth, high market share of products; skinny dog type (Dog), namely a lower sales growth, low market share rate products. Boston matrix was originally used for classical management theory of enterprise research. As a successful management model, it provides very good references in many aspects for research management. But when model is used for analyzing the correlation of the factors in the research management, we need to do in-depth analysis and make the appropriate adjustment and reform in accord with the particularity of scientific research and management activities.

How to connect requirements to your architecture? Liao, Lin. "From Requirements to Architecture: The State of the Art in Software Architecture Design."

The system stakeholders can use software architecture as a basis to understand the system, form consensus, and communicate with each other. The requirements can be classified as functional requirements and non-functional requirements. These requirements are the key input to the software architecture design. As the architect reviews the requirements and proceeds with the design, some modifications to the requirements may be needed. This will allow the architect to communicate with the stakeholders. This way requirements can be connected with the architecture.

Can you connect all requirements directly? What do you do if you cannot? De Boer, Remco C., and Hans Van Vliet. "On the similarity between requirements and architecture."

Not all requirements are connected directly to architecture. Requirements which are at architecture level are only connected to architecture. So the ones which are not at architecture level are not considered. The requirements which are not clear enough or does not contain any information are needed to be detailed so that these can be used for connecting to architecture.

Article: Regnell & Brinkkemper "Market-Driven Requirements Engineering for Software Products"

This article provides an overview of the special characteristics of market-driven requirements engineering and describes the most important challenges of the area. Key elements of market-driven requirements engineering processes are presented together with a definition of process quality. Market-Driven Requirements Engineering (MDRE) covers the classical RE activities, such as elicitation, specification, and validation, adapted to the market-driven situation, where a software producer develops a product that is offered to an open market with many customers. MDRE also covers the specific activities needed in a market-driven context, such as release management and market analysis. MDRE is often conducted under the pressure of competition from other producers, and as the market and product evolve, the MDRE process enacted by a specific software developing organization also needs to be evolved in order to stay ahead of competition. There are a number of variants of software products. (1) The degree of customization and (2) the hardware/software content. A product is said to be generic if it is intended to be used as-is, out-of-the-box, perhaps with minor configurations that are possible to be done by the end-user. A product is said to be customized if the product is intended to be useful after it has been tailored to one specific customer's needs, e.g. through adding modules via an open application interface. A product is said to be customer specific if the entire product is developed with one particular customer's wishes in mind. The hardware/software content is divided into three classes: pure hardware denotes products that are fixed through its hardware architecture and contains no software that can make the features of the product flexible; embedded systems imply products consisting of both a hardware platform and accompanying embedded software; pure software denote a product that is completely comprised of software and sold independently of its hardware platform. Some of the most important characteristics of a typical MDRE context are summarized subsequently.
1. The developing organization makes all decisions but also takes all risks.
2. There is a continuous flow of requirements throughout the product lifetime.
3. The requirements volume is potentially very large and continuously growing.
4. A majority of the requirements are informally described.
5. The product is evolving continuously and delivered in multiple releases.
6. Release planning focuses on time-to-market and return-on-investment

In a survey on market-driven requirements engineering a number of challenges were identified.

1. Balancing market pull and technology push.
2. Chasm between marketing and development.
3. Organizational instability and market turbulence.
4. Simple tools for basic needs.
5. Requirements dependencies.

6. Cost-value-estimation and release planning.
7. Overloaded Requirements Management.

The process quality is intimately related to the quality of the artefacts that are produced during the process. Available resources and lead time until the planned date of the product release into the market limit the realization of any wish into the software product. Market-driven software implies that the vision and scope of the product are well established, thereby setting means to discern whether a requirement fits the standard or is to be rejected as it is too customer specific. In keeping stock of the large volumes of requirements through the stages of the development a requirements state model is indispensable. In order to monitor the progress of the work on the requirements the following statuses of the requirements salmon ladder are usually distinguished. Candidate, Approved, Specified, Discarded, Planned, Developed, Verified and Released. In order to register the requirements properly many development teams use requirements repository.

Website: D. Leffingwell "Scaled Agile Framework"

SAFe is based on number of immutable, underlying Lean and agile principles. These are fundamental tenets, the basic truths and economic underpinnings that drive the roles and practices that make SAFe effective. The nine principles are:

1. Take an economic view
2. Apply systems thinking
3. Assume variability, preserve options
4. Build incrementally with fast, integrated learning cycles
5. Base milestones on objective evaluation of working systems
6. Visualize, reduce batch size and manage queue lengths
7. Apply cadence, synchronize with cross domain planning
8. Unlock the intrinsic motivation of knowledge workers
9. Decentralize decision making

Article: Wnuk et al. "Factors Affecting Decision Outcome and Lead-time in Large-Scale Requirements Engineering"

Lead-time is crucial for decision making in market-driven requirements engineering. In order to identify what factors influence the decision lead time and outcome, a case study was conducted at a large product software manufacturer and statistically analyzed seven possible relationships among decision characteristics. Based on the decision characteristics, five variables were created for each decision.

1. Lead time
2. Number of products affected
3. Release number
4. Type of customer
5. Decision outcome

A survey was conducted among 50 respondents from industry to validate the results from the case study as well as to strengthen the external validity of the study. The survey respondents were mainly working for companies producing product software using the SPL approach. A few hypothesis were made in the study they are:
H1 0: The correlation between the number of products affected by a decision and the lead time needed to take the decision is 0.
H1 1: The correlation between the number of products affected by a decision and the lead time needed to take the decision is not 0.
H2 0: The correlation between the release number of the product line platform attribute of the change requests and the lead time needed to take the decision is 0.
H2 1: The correlation between the release number of the product line platform attribute of the change requests and the lead time needed to take the decision is not 0.
H3 0: The average lead time needed to take a decision is not different when an important customer issues a request.
H3 1: The average lead time to take a decision is different when an important customer issues a request.
H4 0: The number of products affected by a decision is not different for the different decision outcomes.
H4 1: The number of products affected by a decision is different for the different decision outcomes.
H5 0: The release number a decision affects is not different for the different decision outcomes.
H5 1: The release number a decision affects is different for the different decision outcomes.
H6 0: The frequencies in the contingency table between the decision outcome and involvement of important customers do not differ from the normal expected frequencies.
H6 1: The frequencies in the contingency table between the decision outcome and involvement of important customers differ from the normal expected frequencies

The results from the study could be summarized in the following points:

1. The lead time to make a decision increases when more products (considered as a proxy for the decision complexity) are affected by this decision - this result was confirmed both in the statistical analysis and in the survey. Since the relationship is rather clear, decision makers should be aware that too complex decisions may take a long time (hypothesis H1).
2. The statistical analysis showed that if a request affects a lot of products, it has a higher change of being accepted (hypothesis H4). The respondents of the survey stated that requests that affect a lot of products have a higher change of being rejected. This may seem counter-intuitive, but this is probable caused by the fact that request that affect a lot of products are often requests related to the platform and thus are important.
3. There is no significant relationship between the release of the product line that a change request impacts and the decision lead time according to the results from the statistical analysis of the decision log. At the same time, the majority of the

respondents in the survey suggested that decisions made late in the release cycle have shorter lead times (hypothesis H2).

4. Change requests affecting late releases have a significantly higher probability of acceptance according to the statistical analysis of the decision log (hypothesis H5). This result seems to be more characteristic for large contexts as the results from the survey, in which most respondents worked with projects with fewer than 100 decisions, indicate the opposite relationship with a higher probability of rejecting these requests.
5. The lead time for decisions is shorter when the change requests are issued by important customers, according to the respondents (hypothesis H3). The statistical analysis of the decision log disproved this suggestion. Therefore, no clear relationship was identified for this factor.
6. Change requests issued by important customers are more likely to be accepted, (hypothesis H6) according to the statistical analysis of the decision log. This relationship was confirmed by a clear majority of survey respondents (83.3%).
7. The lead time to reject a decision is significantly longer than to accept a decision, according to the statistical analysis of the decision log. At the same time, the results from the survey suggests that there is no relationship between the lead time and the decision outcome.

Seminar 3

Article: Wnuk et al. "What Happened to Our Features? Visualization and Understanding of Scope Change Dynamics in a Large-Scale Industrial Setting"

This paper presents an industrial case study in a large-scale setting where a technique called Feature Survival Charts for visualization of scoping change dynamics has been implemented and evaluated in three projects. Results are based on empirical data from industrial projects at a large company that is using a product line approach. The company has more than 5000 employees and develops embedded systems for a global market. The development of the FSC chart and corresponding scope tracking measures was performed in an interactive manner that involved practitioners from the case company. The persons that participated in the constant evolution and evaluation include one process manager, two requirements managers and one KPI (Key Performance Indicators) manager. This approach involves a set of meetings and discussion points between the researchers and the practitioners that helped to guide the research. The criteria of the project include (1) the length of analyzed project, (2) the number of features considered in the scope of the project and (3) the possibility to visualize and analyze significant scope changes in the analyzed project. An exporter was made to retrieve the data from the scope parameter of each feature in the Feature List document. This information was later sorted so that each feature is mapped into one row and each value of the scope attribute is mapped to an integer value. A meeting with practitioners was held in order to present and discuss results as well as address issues for future work and in the meeting decision was taken to introduce and evaluate a set of scope tracking measurements that may give a better insight into the scoping process practices and may help to assess their quality. The *Feature Survival Chart*, shows scope changes over time which is illustrated on the X-axis. Each feature is positioned on a specific place on the Y-axis so that the complete lifecycle of a single feature can be followed by looking at the same Y-axis position over time. The various scope changes are visualized using different colors. As a result, each scope change can be viewed as a change of the color. The larger the red areas are, the more features are de-scoped in the particular time of the project. Results indicate that in average people experience almost one scope decision per feature for each project. This fact indicates the need for a better understanding of the scoping process, e.g. by visualizing scope changes. A qualitative analysis of the graphs indicates that for all analyzed projects the dominant trend is descoping rather than scope increases. The data was gathered during autumn 2008 when all three projects were running in parallel and were targeted for product releases in 2008 or 2009. Results indicate that we in average experience almost one scope decision per feature for each project. This fact indicates the need for a better understanding of the scoping process, e.g. by visualizing scope changes. A qualitative analysis of the graphs indicates that for all analyzed projects the dominant trend is descoping rather than scope increases. We name this phenomena *negative scoping*. For all analyzed projects we can observe negative scoping all through the analyzed period. Finally it was concluded that visualization technique gives a better overview of the scoping process of the whole project on a single page size graph. The industrial evaluation has indicated that the method can be applied to large scale projects, which demonstrates the scalability of the method.

Seminar 4

Some of the requirements management tools available in the market are:

1. Innoslate- https://www.innoslate.com/
2. Enterprise architect- http://www.sparxsystems.com.au/products/ea/index.html
3. IBM rational Doors- http://www-03.ibm.com/software/products/sv/ratidoor
4. Rally- https://www.rallydev.com/product-feature/requirements-management
5. Version one- https://www.versionone.com/agile-101/agile-project-management-customer-management-best-practices/

Some of the tools for agile requirements management are:

1. Agile manager- http://www8.hp.com/se/sv/software-solutions/agile-project-management-software-development/
2. Blueprint- http://www.blueprintsys.com/
3. CA agile central- http://www.ca.com/us/why-ca/agile-management.html?intcmp=headernav
4. Caliber- http://www.borland.com/en-GB/Products/Requirements-Management/Caliber
5. Cognition cockpit- http://cognition.us/
6. InteGreat- http://www.modernrequirements.com/integreat/
7. Jira software- https://www.atlassian.com/software/jira

Article: Debbiche & Diener "Assessing challenges of continuous integration in the context of software requirements breakdown: a case study"

This paper focuses on issue to identify the challenges of continuous integration and requirements break down and how the latter influences the implementation of a continuous integration process. Through a single case study, 13 semi-structured interviews were conducted and a set of challenges related to the adoption of continuous integration and software requirements breakdown were identified.
In relation to RQ1 - continuous integration adoption challenges, the most dominant results include: 1) the mindset is an important factor in the success of implementing continuous integration. Skepticism towards the introduction of a new process needs be considered in order to win over non-believers. 2) Testing tools and the maturity of the infrastructure supporting the continuous integration process is required in order to facilitate the daily tasks involved. Continuous integration advocates the use of automated tools to allow more frequent and efficient integrations. 3) Similar to Agile, assumptions made by the concept of continuous integration may not apply to all organizations, products and projects, especially those of larger dimensions. 4) Lacking and under-communicating a clear goal can delay and cause confusion among developers when transitioning to continuous integration.
Some of the identified challenges such as mindset, tools and infrastructure maturity and testing have been mentioned in previous literature when transitioning to continuous integration. However, this study also identified software requirements as a challenge when adopting continuous integration.
The findings regarding RQ2 - software requirements breakdown challenges, the most prominent results are: 1) Requirements abstraction plays a deciding factor in the success of breaking down software requirements. Requirements that are too large or ambiguous complicates the matter of finding the right level of abstraction of a requirement. 2) Breaking down software requirements to allow more frequent integrations makes it hard to guarantee delivery of customer value since some tasks or units of requirements might not seem like a benefit to the customer. That said, keeping both the customer and the implementation perspective in mind while breaking down software requirements is important. 3) Providing a clear guiding principle is important in order to facilitate the breakdown of requirements for the teams. Though the focus of this research question is on the challenges of software requirements breakdown, they are inherently contextualized by the adoption of continuous integration.
When it comes to RQ3 - software requirements breakdown's influence on continuous integration, the majority of the discussion includes: 1) The need of breaking down software requirements in order to allow more frequent code integrations correlates with the sought after integration frequency. 2) Breaking down software requirements implies finding a balanced integration scope as well as considering test and implementation dependencies. However, keeping a flexible quality assurance policy of the product branch might increase

the possibilities of breaking down software requirements with regards to continuous integration.
Finally it was concluded that the role of software requirements and their breakdown plays an important role when adopting continuous integration, especially with regards to an increased integration frequency.

Reflections:

The articles by Ruhe, Berntsson Svensson & Olsson and R: Berntsson Svensson, B. Regnell given in the seminar 0 discusses about the release planning, QUPER model which were helpful in knowing the RE concepts much better and were useful in writing the summary of release planning and were helpful in answering other questions in seminar 0. Article by Regnell gave initial knowledge on the market driven requirements engineering process which helped me to follow on to Regnell & Brinkkemper "Market-Driven Requirements Engineering for Software Products" article in seminar 2. Articles in seminar 1 discusses about the concept of requirements abstraction and prioritization by introducing different methods and models which were helpful in writing my reflective report assignment (Gorschek & Wohlin "Requirements Abstraction Model" where the requirements are placed into different levels and prioritized later). (Khurum & Gorschek "A method for early requirements triage and selection utilizing product strategies" where the requirements are given weightage and normalized data is calculated to identify the priority order of requirements). GAP analysis was described by Tony Gorschek & Davis in their paper "Requirements Engineering. In search of dependent variables." Which allowed me find more related literature from the databases. Regnell & Brinkkemper "Market-Driven Requirements Engineering for Software Products" and v.d. Weerd & Brinkkemper "Towards a reference framework for software product management" provided more information on market driven requirements engineering and were helpful in answering the road mapping and technical product management questions in seminar 1. Wnuk et al. "What Happened to Our Features? Visualization and Understanding of Scope Change Dynamics in a Large-Scale Industrial Setting" from the seminar 3 shows the case study in a large scale setting feature survival charts (FSC) technique allowed me to learn the technique with the help of the article.