Fraunhofer Institut
Experimentelles
Software Engineering

Quantitative WinWin – A New Method for Decision Support in Requirements Negotiation

Authors:
Günther Ruhe
Armin Eberlein
Dietmar Pfahl

Accepted for publication in
Proceedings of SEKE 2002,
Ischia, Italy, July 15-19, 2002

IESE-Report No. 014.02/E
Version 1.0
April 5, 2002

A publication by Fraunhofer IESE

Fraunhofer IESE is an institute of the Fraunhofer Gesellschaft.
The institute transfers innovative software development techniques, methods and tools into industrial practice, assists companies in building software competencies customized to their needs, and helps them to establish a competetive market position.

Fraunhofer IESE is directed by
Prof. Dr. Dieter Rombach
Sauerwiesen 6
D-67661 Kaiserslautern

Abstract

Defining, prioritizing, and selecting requirements are problems of tremendous importance. In this paper, a new approach called Quantitative WinWin for decision support in requirements negotiation is studied. The difference to Boehm's WinWin groupware-based negotiation support is the inclusion of quantitative methods as a backbone for better and more objective decisions. Like Boehm's original WinWin, Quantitative WinWin uses an iterative approach, with the aim to increase knowledge about the requirements during each iteration. The novelty of the presented idea is three-fold. Firstly, it uses the Analytical Hierarchy Process for a stepwise determination of the stakeholders' preferences in quantitative terms. Secondly, these results are combined with methods for early effort estimation, in our case using the simulation prototype GENSIM, to evaluate the feasibility of alternative requirements subsets in terms of their related implementation efforts. Thirdly, it reflects the increasing knowledge gained about the requirements during each iteration, in a similar way as it is done in Boehm's spiral model for software development. As main result, quantitative WinWin offers decision support for selecting the most appropriate requirements based on the preferences of the stakeholders, the business value of requirements and a given maximum development effort.

Keywords: Requirements negotiation, decision support, quantitative methods, analytical hierarchy process, effort estimation, simulation, Easy WinWin

Table of Contents

1 Background and Motivation

Software development is a very complex, often distributed process with a variety of process and product attributes, as well as numerous functional and non-functional objectives and constraints. Brooks [5] describes the importance of selecting and describing the right requirements: "The hardest single part of building a software system is deciding precisely what to build." Typically, requirements are imprecise or incomplete at the beginning of the life cycle and are becoming more and more complete and precise as development progresses. A further complication is that the various stakeholders have different perspectives of and expectations on a system, i.e., they assign varying priorities to requirements. Marketing and project managers also struggle with a lack of a clear understanding of the relationship between a selected set of requirements and the effort required for its implementation. The overall effort that can be spent on the project is limited and the time to market of the final product is constrained. The challenge is to select the 'right' requirements out of a given superset of candidate requirements so that all the different key interests, technical constraints and preferences of the critical stakeholders are fulfilled and the overall business value of the product is maximized.

Development support for software projects currently available includes numerous design tools, configuration management tools and requirements management tools. Over the last few years, these tools have helped improve software quality and are now in wide spread use in industry. However, they are domain-independent, i.e., they cannot provide domain-specific support. Furthermore, while many of these tools have been successful in providing support for modeling and code generation, they have been less successful in supporting requirements analysis [9].

Domain-specific support is difficult to provide. The requirements engineering community has developed some research tools [7], [13] that provide domain-specific assistance. However, such tools are not widely used in industry. The main reasons for this are the effort required to initially model the domain in sufficient detail as well as the effort required for keeping the domain models up-to-date. Any changes at higher levels of abstraction would cause major changes in the domain models. Considering the rapid changes in today's domains, this approach appears to become less and less feasible.

The distributed nature of system development has resulted in several approaches that provide collaboration support. The most-common ones are telephone and video conferencing facilities. Various groupware systems are also in use. Boehm et al. [2] have created a prototype for his WinWin spiral model, a

tool that assists stakeholders in creating win-win solutions for all parties involved. Furthermore, research is being done in the general area of computer support for decision-making groups involved in the negotiation of requirements over a distance. Damian et al. particularly investigate the effect of the communication media (audio, video, etc.) on the negotiation outcome [6].

Unfortunately, there is so far no comprehensive support for making decisions throughout software development. Such decision support would be most helpful when decisions have to be made in complex and dynamic environments. Support here means assistance for structuring the problem, for analyzing and verifying the obtained structure, as well as help for comparing the results of different choices using simulation. Decision makers need support to describe, evaluate, sort, rank, select or reject candidate products, processes or tools.

In this paper, a new approach called Quantitative WinWin for decision support in requirements negotiation is studied. The difference to Boehm's WinWin groupware-based negotiation support is the inclusion of quantitative methods. Like Boehm's original WinWin, Quantitative WinWin uses an iterative approach, with the aim to increase knowledge about the requirements in each during all iteration. A key feature of the quantitative approach is the elicitation of implicit preferences of involved stakeholders using Analytical Hierarchy Process (AHP). The combination of AHP with effort estimation using modeling and simulation, as well as the stepwise inclusion of requirements into the requirements subset to be implemented provides comprehensive assistance for the selection of optimal requirements with respect to the trade-off between business value and development effort.

The paper is organized as follows: Section 2 provides definitions and assumptions as well as a mathematical description of the problem. Section 3 explains the methodological underpinnings used in our approach, namely the incremental treatment of requirements, the use of Analytic Hierarchy Process (AHP) to assign priorities to stakeholders and the different requirements classes, as well as the computation of optimal trade-offs for subset selection under resource constraints. Section 4 presents the simulation model GENSIM that is used in section 6 for effort estimation and an initial validation of the Quantitative WinWin approach considering three stakeholders and three classes of requirements. The overall quantitative WinWin approach is described in detail in section 5. Finally, a summary and conclusions are given at section 7.

2 Underlying Assumptions and Problem Statement

2.1 Classes of Stakeholders and Requirements

For the purpose of this paper we assume a set $\Re = \{r_1,\ldots, r_n\}$ of requirements $r_1,\ldots, r_n$. The set $\Re$ is subdivided into q disjoint subclasses $R_1, R_2,\ldots, R_q$ (q<n) of requirements, e.g., $\Re = R_1 \oplus R_2 \oplus \ldots \oplus R_q$. The rational for this assumption is that requirements can be classified according to their resource needs, their interaction (two requirements interact if and only if the satisfaction of one requirement affects the satisfaction of the other [14]) and their purpose. Each class contains requirements of the same type. Example classes are requirements related to the user interface, or non-functional requirements like reliability or maintainability. Each requirement r_i is assumed to belong to exactly one class. Mapping $\psi(i)$ assigns each requirement $r_i \in \Re$ to the associated class $R_{\psi(i)}$.

Another important assumption addresses the importance of requirements. Some of the given requirements are mandatory and related to the core functionality of the system. Others are more or less optional. We assume that each requirement r_i has a relative importance $\alpha(r_i) \in (0,1]$ within class $R_{\psi(i)}$. That is defined on a rational scale. $\alpha(r_i) = 1$ means that the requirement is mandatory. On the other hand, $\alpha(r_i)=2\alpha(r_j)$ means that requirement r_i is twice as important than requirement r_j. These relative importance factors within each class are assumed to be objective, i.e., independent of the different stakeholder perspectives.

One of the challenges of requirements engineering is that the different stakeholders of a system usually have conflicting viewpoints and preferences. To reflect and model this situation, we introduce different classes of stakeholders. The p classes are abbreviated by $S_1, S_2,\ldots, S_p$. Later we will use the Analytical Hierarchy Process (AHP) to elicit the stakeholders' preferences regarding the different classes of requirements (not the individual requirements contained in the classes). In the worst case, these preferences can still change over time. This issue is addressed by the iterative nature of our overall approach.

Example stakeholders are novice, advanced or expert users as used in the example described in section 6. Other stakeholders can be e.g., managers (having a business-driven perspective) or developers (having a more technical perspective). In any case, different classes of stakeholders will have different objectives and ideas for developing, using and handling the final software product.

2.2 Problem Statement

The original set of requirements $\Re$ is considered to be a superset containing the requirements of all stakeholders involved. Because resources are limited, the challenge is to find those subsets of requirements that can be implemented without exceeding the given effort maximum. For that reason, we define an upper effort bound denoted by EFFORT. We then need to assign to each subset $\Re^* \subset \Re$ an estimated value called ef($\Re^*$) which represents the estimated effort required for implementing the requirements subset $\Re^*$. We can now formulate our implementation effort constraints as ef($\Re^*$) $\leq$ EFFORT.

Typically, estimates for ef($\Re^*$) are difficult to get and uncertain in their nature. Example estimation methods are (adjusted) function points [1], or simulation methods, such as system dynamics when combined with expert judgment during modeling [11]. For instantiation and illustration of the generic solution approach as presented in section 4, we assume that there is a simulation model based on system dynamics that is able to provide appropriate effort predictions.

In order to be able to select requirements subsets that maximize the business value in relation to its necessary implementation effort, the importance of the different classes of requirements from the perspective of the different stakeholders has to be determined. From the application of AHP, we assume a normalized vector of importance $\beta = (\beta(R_1), \beta(R_2), \ldots, \beta(R_q))$ of the q classes of requirements. The absolute importance $\chi(r_i)$ of each requirement r_i is the product of the importance of the associated class $\beta(R_{\psi(i)})$ with the relative importance $\alpha(r_i)$ of requirement r_i within class $R_{\psi(i)}$ i.e., $\chi(r_i) = \alpha(r_i) * \beta(R_{\psi(i)})$. With the notation introduced above we are now able to describe the problem "Requirements Subset Selection (RSS)" in a more formal way:

Requirements Subset Selection (RSS): Find all subsets of requirements $\Re^* \subset \Re$ such that

(1) ef($\Re^*$) $\leq$ EFFORT,
(2) $\Re^*$ is effort maximal , i.e., if one element is added to $\Re^*$ then this would violate (1), and
(3) (ef($\Re^*$), $\Sigma_{r \in \Re^*} \chi(r)$) is a non-dominated solution, i.e., there is no other effort maximal set with less effort and at least the same business value or with more business value and at least the same effort.

In the following, we will describe the methodological prerequisites to solve RSS.

3 A Hybrid and Quantitative Approach to Support Requirements Subset Selection

This section describes a new approach for stepwise requirements subset selection called Quantitative WinWin. The overall method for solving RSS consists of three main components that are described in more detail in the subsequent parts of this section:

- Analytic Hierarchy Process (AHP)
- Stepwise relaxation of the required importance level of candidate requirements
- Optimal trade-offs for requirements subset selection under given resource constraints

The combination of these techniques helps to find subsets of requirements that represent optimal trade-offs between effort and overall business value. As a result, the actual decision maker can finally choose from a set of pair wise incomparable points of a trade-off curve (see also the validation example and Fig 6).

The novelty of the presented idea is three-fold. Firstly, it uses AHP for a stepwise determination of the stakeholders' preferences in quantitative terms. Secondly, these results are combined with methods for evaluating the feasibility of selected subsets of requirements in terms of their implementation effort. Thirdly, it reflects the increasing degree of knowledge gained about the requirements in each iteration in the same way as it is done in Boehm's spiral model [3] for software development. Even though inherently difficult, we assume that an effort estimation method can be applied based on a given specification of requirements. As discussed in [4], estimation is increasingly based on a combined use of expert opinion and simulation. In section 4, we describe how the simulation model GENSIM (GENeric SIMulator) is used for the purpose of effort estimation. The GENSIM model allows the simulation of the software development process from the end of the requirement analysis step down to the end of system testing.

3.1 Analytic Hierarchy Process

The Analytic Hierarchy Process (AHP) [15] is a systematic approach to elicit implicit preferences between different attributes. For the purpose of this investigation, AHP is applied to determine the importance of the various stakeholders

from a business perspective. In addition, it is used to prioritize the different classes of requirements from the perspective of each stakeholder. The two preference schemata are combined to rank the importance of the different classes of requirements for the final business value of the software product. The outcome of the two-step AHP analysis is used to determine most preferred (i.e., non-dominated) solutions R* at the τ-th iteration which is reached when the given effort constraint is violated for the first time.

AHP assumes that the problem under investigation can be structured as an attributive hierarchy with at least three levels. At the first level, the overall goal is described. The second level describes the different competing criteria that refine the overall goal of level 1. Finally, the third level is used for the selection from competing alternatives.

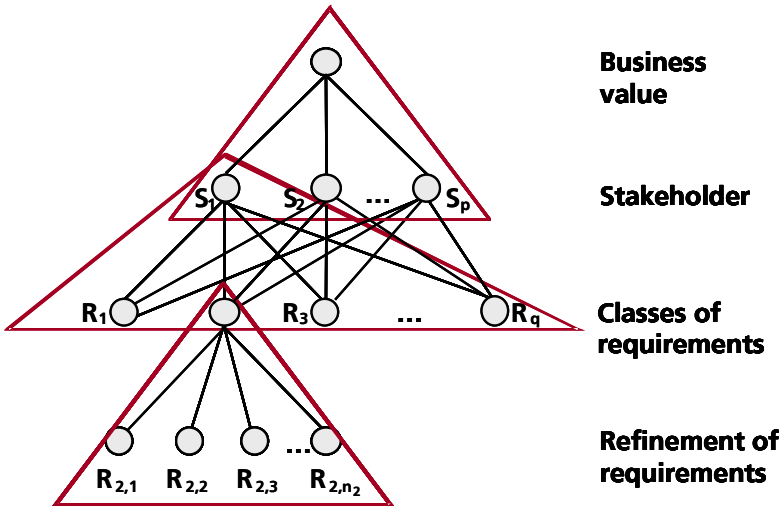

Figure 1: Example AHP graph for optimal requirements selection.

In the context of this paper, the first level corresponds to the maximum business value of the final software product. In general, there are p nodes in the second level that correspond to the p stakeholders. In the same way, the q nodes at the third level are associated with the q classes of requirements assumed. If necessary, a fourth level can be introduced used for a further refinement of the classes of requirements. The triangles indicate the way attributes of one level are assessed from the perspective of the parent node. This is shown in Figure 1.

At each level of the hierarchy, a decision-maker performs a pair-wise comparison of attributes assessing their contributions to each of the higher level nodes to which they are linked. This pair-wise comparison involves preference ratios (for actions) or importance ratios (for criteria). The expert assigns an importance number that represents the importance of a term t_i with respect to an-

other term t_j to describe the domain. Saaty [15] introduced a nine-point intensity scale:

- Of equal importance → preference number=1
- Of moderate importance → preference number=3
- Of strong importance → preference number=5
- Of very strong importance → preference number=7
- Of extreme importance → preference number=9.

To get a ranking of priorities from a set of attributes, the best approach is the eigen-value solution [15]. From iterative application of this procedure, we receive the weights of importance of each requirement of the domain with respect to the overall business goal. There exist commercially available tools that compute the eigen-values and check the degree of consistency between the pair-wise comparisons. For the computations of the example in section 6, we used a tool called ExpertChoice [17].

3.2 Incremental Refinement of Requirements Selection

An iterative approach for the incremental refinement of requirements and resource constraints is chosen to reflect the impreciseness of requirements and their dynamically changing character. The motivation for the stepwise inclusion of more requirements is two-fold.

Firstly, requirements are typically uncertain at the beginning, and become more and more precise during the software life cycle. This is explicitly assumed in the spiral software development model, but is valid to some extend also for other software development paradigms. Consequently, requirements selection has to become increasingly precise. At the beginning, the focus is on the most important requirements, i.e., those requirements with largest value $\chi(r_k)$. Later, conditions are gradually relaxed to include as many requirements with lower importance as possible until the maximum effort bound is reached.

Secondly, as another indication of incompleteness and impreciseness of requirements, it may happen that new requirements are added during (spiral) software development. This is reflected by the iterative approach where you can consider additional requirements for inclusion that appeared at a later stage of development.

The idea is to consider a sequence $\{RSS^i\}_{i=1,\ldots,s}$ of problems of type RSS. However, only the final one is solved explicitly. The different problems are characterized by different sets of requirements $\{\mathfrak{R}^i\}_{i=1,\ldots,s}$. Assume there is a monotonously decreasing sequence $\{level^i\}_{i=1,\ldots,s}$ of levels of minimum importance $\chi(r_k)$. The set $\mathfrak{R}^i$ is defined as the set of requirements with weighted importance of

being at least level^i, i.e., $\Re^i = \{r_j \in \Re: \chi(r_j) \geq \text{level}^i\}$. This implies that $\Re^1 \subseteq \Re^2 \subseteq \ldots \subseteq \Re$. The number τ of iterations is defined as the first iteration at which the total amount of effort would exceed the specified maximum value EFFORT, i.e. too many candidate requirements have been selected for implementation. In this case, the approach described in section 3.3 for requirements subset selection under given resource constraints is applied.

3.3 Optimization of Requirements Selection

We assume that at the τ-th iteration the given effort constraint is violated for the first time, i.e., $ef(\Re^\tau) > \text{EFFORT}$ and $ef(\Re^{\tau-1}) \leq \text{EFFORT}$. Problem $\{RSS^\tau\}$ is a discrete optimization problem with a non-linear and non-additive effort function ef: $\Re \Rightarrow ef(\Re)$ that cannot be formulated explicitly. We use an enumeration algorithm to determine those sets of requirements that maximize the effort and are non-dominated. A solution with two objectives (in our case, effort and business value) is called non-dominated if there is no other (feasible) solution that is strictly better in one criterion and not worse in the other one. In our case, the set of non-dominated solutions is abbreviated by Γ. To compute Γ we systematically eliminate requirements from the infeasible set $\Re^\tau$ and check the feasibility of the resulting set. Finally, a consistency check within Γ is conducted to make sure that no solution within Γ is dominated by another one. The recursive procedure with input set $\Re^\tau$ can now be described as follows:

Algorithm RSS($\Re^\tau$)
begin
$\Gamma = \varnothing$
for all $r_k \in \Re^\tau$ **do**
begin
if $ef(\Re^\tau - \{r_k\}) \leq \text{EFFORT}$
then $\Gamma = \Gamma +$ $\{\Re^\tau - \{r_k\}\}$ **else** $RSS(\Re^\tau - \{r_k\})$
end
for all $\Re^* \in \Gamma$ **do**
begin
if there is a feasible set $\Re\text{^} \in \Gamma$ such that
either $ef(\Re\text{^}) < ef(\Re^*)$ & $\Sigma_{r \in \Re\text{^}} \chi(r)) \geq \Sigma_{r \in \Re^*} \chi(r)$
or $ef(\Re\text{^}) \leq ef(\Re^*)$ & $\Sigma_{r \in \Re\text{^}} \chi(r)) > \Sigma_{r \in \Re^*} \chi(r)$
then $\Gamma = \Gamma - \Re^*$
end
end

4 Effort Estimation using "GENSIM"

In order to generate estimates of ef(ℜ) we used the simulation model GENSIM (GENeric SIMulator) [12]. As has been mentioned before, the GENSIM model can be used for the simulation of the software development process from the end of the requirement analysis step through to the end of system testing (cf. Figure 2). Although the model is only a research prototype it can be easily calibrated to product and process measures of a specific organization in order to capture the behaviour of each development cycle in Boehm's spiral model. The model has successfully been used for the simulation of processes in various companies. For producing the effort estimates used in the example in section 6 of this paper, GENSIM was calibrated to the development process of a fictitious software organization. The GENSIM model has a modular structure. It consists of five interrelated sub-models:

- **Production:** This sub-model represents a typical software development cycle consisting of the following steps of transitions: set of requirements → design documents → code → tested code. Note that the detection of defects during testing only causes reworking of the code (and not of the design documents).
- **Quality:** In this sub-model, the defect co-flow is modeled, i.e.: defect injection (into design or code) → defect propagation (from design to code) → defect detection (in the code during testing) → defect correction (only in the code).
- **Effort:** In this sub-model, the total effort consumption for design development, code development, code testing, and defect correction (rework) is calculated.
- **Initial Calculations:** In this sub-view, the normal value of the central process parameter "productivity" is calculated. The normal productivity varies with assumptions about the product development mode (organic, semi-detached, embedded) and characteristics of the project resources available (e.g. developer skill).
- **Productivity, Quality & Manpower Adjustment:** In this sub-model, project-specific process parameters, like (actual) productivity, defect generation, effectiveness of QA activities, etc., are determined based on a) planned target values for manpower, project duration, product quality, etc., and b) time pressure caused by unexpected rework or changes in the set of requirements.

Input Parameter	Output Parameter
Product_size [total number of size units]	Design_size [total number of designed and inspected size units]
Average_complexity [1 = low, 3 = medium, 5 = high]	Code_size [total number of implemented and inspected size units]
Manpower_skill [1 = low, 2 = medium, 3 = high]	Product_size [total number of implemented and tested size units]
Planned_manpower (optional) [number of persons]	Project_duration (project total and per phase) [weeks]
Planned_completion_time (optional) [weeks]	Effort (project total and per phase) [person weeks]
Goal_field_defect_density (optional) [defects per implemented size unit]	Field_defect_density [defects per implemented size units after test]
Inspection_intensity_design [fixed percentage of total number of size units]	
Inspection_intensity_code [fixed percentage of total number of size units]	

Table 1: Input and output parameters of the GENSIM model.

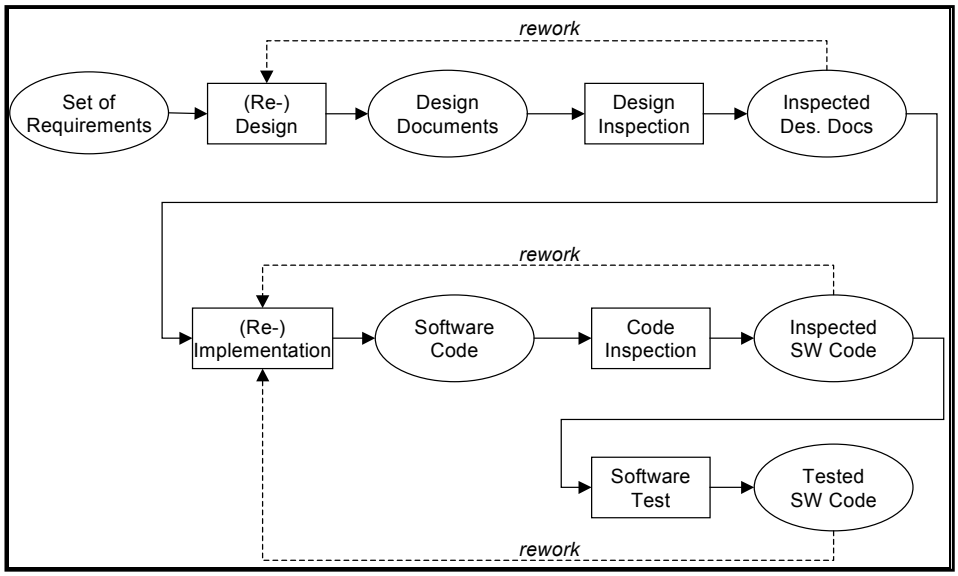

Figure 2: Schematic representation of the product flow captured by the GENSIM production sub-model.

A detailed description of the GENSIM model can be found in . The most important input and output parameters are listed in Table 1. The input parameters of the simulation define the project goals (Product_size, Planned_completion_time, Goal_field_defect_density) and constraints (Average_complexity, Planned_manpower, Manpower_skill), as well as the process, e.g. the degree

to which design and code inspections are applied (Inspection_intensity_design, Inspection_intensity_code). The output parameters represent the simulation results, e.g., size of the work and end products (Design_size, Code_size, Product_size), project duration (Project_duration), effort consumption (Effort), and product quality (Field_defect_density). For the calculations conducted in the example presented in section 6, all the input (and output) parameters with grey shading have been varied as part of the simulation runs.

The simulation modeling approach used to develop GENSIM has been defined in [12] under the name IMMoS (Integrated Measurement, Modeling and Simulation). IMMoS is an enhancement and operationalization of the well-known System Dynamics method, originally developed by Forrester in the late 1950s [8]. The philosophical position underlying the System Dynamics method is what Senge and other researchers call System Thinking [16]. In System Thinking, the behavior of a system is considered as primarily being generated by the interaction of all the feedback loops over time. In order to analyze – and eventually change – the behavior of observed objects in the real world, it is necessary to understand the important cause-effect relations of the factors that influence those variables that represent the observed behavior. In System Dynamics, these cause-effect relations are called base mechanisms. The union set of all base mechanisms is called a causal diagram. In order to be able to run System Dynamics simulations the causal diagram has to be converted into a so-called flow graph. A flow graph is the pictorial representation of a set of mathematical equations. The set of mathematical equations can be separated into two groups: level equations and rate equations. The terminology of levels and rates is consistent with the flow-structure orientation introduced by Forrester together with schematic conventions invoking the image of fluid-like processes.

Even though System Dynamics has an increasing number of applications in the software engineering domain, it is by no means suggested to be the new silver bullet technique for problem solving. Instead, it is important to clarify the underlying assumptions for System Dynamics modeling and simulation. Only if these assumptions are valid, it is recommended to use the System Dynamics approach in a particular situation. The basic assumptions are:

- Problems under investigation are dynamic in nature and relate to systems with entities and attributes that are interconnected in loops of information feedback and circular causality.
- Sufficient maturity and stability of the software development processes in place in the organization, e.g., CMM level 3 or higher [10].
- Availability of expertise for identification of base mechanisms and construction of causal diagrams.
- Availability of data for model calibration.

These assumptions are valid for the requirements engineering process of most mature software organizations. Requirements are very volatile and prone to numerous changes. If we assume a fairly mature software organization, the software development processes are well thought-through and reasonably stable, and thus hypotheses about base mechanisms should not be too difficult to elicit from experienced project managers – as long System Dynamics experts are available to conduct interviews and transform their input into causal diagrams. Mature organizations are also likely to have a metric collection process in place, i.e., there should be sufficient data available to calibrate the simulation models.

5 Quantitative "WINWIN" – The Overall Algorithm

The algorithm called Quantitative WinWin uses iterative and hybrid application of the techniques described in section 3 and 4. We distinguish six consecutive steps, which are conducted in each iteration i as described in Figure 3. The number of iterations is not determined in advance. It depends on the degree of change in requirements during the development cycle and is finally determined by the expert using Quantitative WinWin for support in negotiating requirements.

Step 1: Definition of the candidate set of requirements.

At the beginning of each iteration i, the threshold value $level^i$ defines a requirements subset that contains those requirements r_j of the original set $\Re$ that have a importance value of at least $level^i$, i.e., $\Re^i = \{r_j \in \Re: \chi(r_j) \geq level^i\}$. The threshold values are not defined in advance and have to be determined by experts. The threshold value will determine the size of the set of requirements under investigation. The only assumption is that $level^{i+1} > level^i$ for all iterations i. An other modification of the candidate set of requirements can be done by adding to $\Re$ further requirements at later iterations (as can be seen in the example in section 6).

Step 2: Computation of preferences between involved stakeholders.

Preferences are computed from the perspective of the overall business value. AHP is applied for that purpose (based on the nine-point scale introduced in section 3.1) resulting in a normalized vector of weights $weight^0 = (weight_{0,1},\ldots, weight_{0,p})$ with $\Sigma\ weight_{0,j} = 1$.

Step 3: Computation of preferences between requirements classes.

Preferences are computed from the perspective of the individual stakeholders. AHP is applied for that purpose (based on the nine-point scale introduced in section 3.1) resulting in normalized vectors $weight^1,\ldots, weight^q$.

Step 4: Computation of overall preferences between requirements classes.

Computation of overall preferences between requirements classes by consecutive application of the weights computed in step 3 (the vectors $weight^1,\ldots,$

weight^q are arranged as the rows of a matrix M) and step 2 (column vector) by multiplication of matrix M with vector weight^0. The result is a vector of importance $\beta = (\beta(R_1), \beta(R_2),\ldots, \beta(R_q))$ of the q classes of requirements.

Step 5: Refinement of candidate requirements.

Subset $\Re^i = \{ r_j \in \Re: \chi(r_j) \geq level^i\}$ with $\chi(r_i) = \alpha(r_i)*\beta\ (R_{\psi(i)})$ is defined where $\Re = \Re + \Delta\Re^i$ is the original set of requirements eventually extended by requirements $\Delta\Re^i$ added in a later stage.

Step 6: Effort estimation and feasibility check.

For each subset $\Re^i = \{ r_j \in \Re: \chi(r_j) \geq level^i\}$ with $\chi(r_j) = \alpha(r_j)*\beta(R_{\psi(j)})$, an effort estimation is done by computing $ef(\Re^i)$. The set of requirements is feasible if and only if $ef(\Re^i) \leq$ EFFORT. If the set is not feasible, the procedure terminates by application of procedure **RSS(**$\Re^i$) for computing the set Γ of non-dominated solutions.

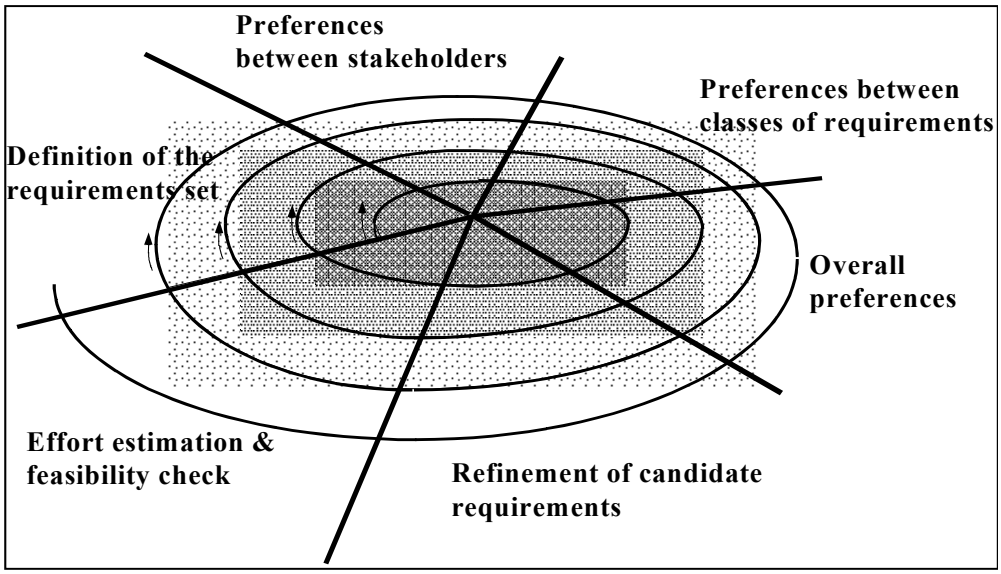

Figure 3: Principal steps of Quantitative WinWin.

6 Validation of the Approach using "GENSIM"

We validate the proposed approach by considering the development of a software product that has three classes of users: novice (S_1), advanced (S_2), and expert (S_3) users. Typically, for a product, e.g., a text processing software system, different classes of users have different requirements on the key features of that product. We assume an original set of ten requirements where each requirement belongs to one of the three classes of requirements R_1, R_2, and R_3. As the project progresses, two additional requirements r_{11} and r_{12} arise. The three classes correspond to three different categories of requirements:

- Performance (class R_1)
- Usability (class R_2)
- Security and reliability (class R_3)

We use GENSIM as introduced in section 4 to model and simulate this example. Basic information about the example and the results of three iterations are summarized in Table 2. Initially, ten requirements are taken into account. Later, two more requirements (r_{11} and r_{12}) are added. Columns $\alpha(r_j)$, $\beta(R_{\psi(j)})$, and $\chi(r_j)$ describe the relative importance of requirement r_j within class $R_{\psi(j)}$, importance of class $R_{\psi(j)}$, and global importance of requirement r_j, respectively. The two final rows of the table give the effort estimation $ef(\Re^i)$ of set $\Re^i$ and the business value $\Sigma_{r \in \Re i} \chi(r)$ associated with $\Re^i$.

The total effort available to implement the final set of requirements is EFFORT = 4300 person days. The development process, manpower size and experience are fixed, as well as the product quality. Each requirement is assumed to have a fixed size and specific complexity (e.g., low, medium or high). Simulation gives us an estimate of the total effort required for the implementation of each subset of requirements.

We assume the product manager has to decide which of the initially given ten (later, this will be twelve) requirements will be selected to maximize the overall business value (step 1). For that purpose, s/he defines a 3x3-matrix M^0 of preferences between the three types of users in terms of the overall goal to maximize the business value. Matrices M^1, M^2, and M^3 describe the preferences of the three classes of requirements (i.e., R_1, R_2, R_3) from the perspective of stakeholder S_1, S_2, and S_3, respectively (see Figure 1). This may look as follows:

$$M^0 = \begin{bmatrix} 1 & 3 & 6 \\ 1/3 & 1 & 4 \\ 1/6 & 1/4 & 1 \end{bmatrix} \qquad M^1 = \begin{bmatrix} 1 & 2 & 5 \\ 1/2 & 1 & 4 \\ 1/5 & 1/4 & 1 \end{bmatrix},$$

$$M^2 = \begin{bmatrix} 1 & 1/3 & 1/6 \\ 3 & 1 & 1/2 \\ 6 & 2 & 1 \end{bmatrix}, \text{ and } M^3 = \begin{bmatrix} 1 & 1/5 & 1/2 \\ 5 & 1 & 4 \\ 2 & 1/4 & 1 \end{bmatrix}.$$

The AHP analysis gives us vectors of eigen-values. They are denoted by weight^0, weight^1, weight^2, and weight^3, respectively.

- weight^0 = (0.644, 0.271, 0.085) gives the importance of the three stake-holders (i.e., novice, advanced, expert) for the final business value from the perspective of the product manager (step 2).
- weight^1 = (0.570, 0.333, 0.097) gives the importance of the three classes of requirements (i.e., R_1, R_2, R_3) from the perspective of the novice user (step 3.1).
- weight^2 = (0.100, 0.300, 0.600) gives the importance of the three classes of requirements from the perspective of the advanced user (step 3.2).
- weight^3 = (0.117, 0.683, 0.200) gives the importance of the three classes of requirements from the perspective of the expert user (step 3.3).

The consecutive application of the weights computed in steps 2 and 3 results in $\beta = (\beta(R_1), \beta(R_2), \ldots, \beta(R_3))$ which is the vector of importance of the three classes of requirements (i.e., R_1, R_2, R_3) for the overall business value (step 4):

$$\begin{array}{l} (R_1) \\ (R_2) \\ (R_3) \end{array} \begin{bmatrix} 0.570 & 0.333 & 0.097 \\ 0.100 & 0.300 & 0.600 \\ 0.117 & 0.683 & 0.200 \end{bmatrix} \begin{bmatrix} 0.644 \\ 0.271 \\ 0.085 \end{bmatrix} \begin{array}{l} (S_1) \\ (S_2) \\ (S_3) \end{array} = \begin{bmatrix} 0.466 \\ 0.188 \\ 0.278 \end{bmatrix} \begin{array}{l} (\beta(R_1)) \\ (\beta(R_2)) \\ (\beta(R_3)) \end{array}$$

According to the individual scorings, the first class of requirements is most important, and the second class is the least important one. Eventually, these scores could be changed as part of the stepwise refinement approach. However, we will not consider such changes in this example.

To illustrate the concept of stepwise refinement, we assume three iterations. At each iteration, the requirements acceptance threshold is increased. In order to simplify the example, we assume that the preference in AHP described by the four 3x3-matrices does not change over time. As stated above, this is not a necessary prerequisite for the applicability of Quantitative WinWin.

Subset characteristics	r_i	Class $R_{\psi(i)}$	$\alpha(r_i)$	$\beta(R_{\psi(i)})$	$\chi(r_i)$	$\Re^1$	$\Re^2$	$\Re^3$	$\Re^{3\setminus r9}$	$\Re^{3\setminus r5}$
	1	R_1	0.6	0.466	0.28	X	X	X	X	X
	2	R_1	0.9	0.466	0.42	X	X	X	X	X
	3	R_1	1.0	0.466	0.47	X	X	X	X	X
	4	R_1	0.7	0.466	0.33	X	X	X	X	X
	5	R_2	1.0	0.188	0.19			X	X	-
	6	R_2	0.6	0.188	0.11					
	7	R_2	1.0	0.188	0.19			X	X	X
	8	R_2	0.5	0.188	0.09					
	9	R_3	1.0	0.278	0.28	X	X	X	-	X
	10	R_3	0.3	0.278	0.08					
	11	R_3	0.8	0.278	0.22	n/a	X	X	X	X
	12	R_3	1.0	0.278	0.28	n/a	n/a	X	X	X
level[1]						0.25				
level[2]							0.20			
level[3]								0.15	0.15	0.15
ef($\Re^i$)						3017	3413	4886	4050	4225
$\Sigma_{r\in \Re i}\ \chi(r)$						1.78	2.00	2.66	2.38	2.47

Table 2: Basic information about the example and the sequence of iterations.

Iteration 1:

The initial requirements acceptance threshold level $level^1$ is assumed to be 0.25. This results in the first requirements subset $\Re^1 = \{ r_j \in \Re: \chi(r_j) \geq level^1\} = \{r_1, r_2, r_3, r_4, r_9\}$. The estimated effort $ef(\Re^1)$= 3017 < 4300=EFFORT is below the given effort bound.

Iteration 2:

In the second iteration, we have a relaxed level of importance of $level^2$ = 0.20. Furthermore, an additional requirement r_{11} is considered to be included into the requirements set. Following the procedure, we get our new set $\Re^2 = \{ r_j \in \Re: \chi(r_j) \geq level^2\} = \{r_1, r_2, r_3, r_4, r_9, r_{11}\}$ which still does not exceed the assumed effort bound ($ef(\Re^2)$= 3413 < 4300=EFFORT).

Iteration 3:

In the third iteration, we again have to add a previously neglected requirement r_{12}. In addition to that, we further relax the required level of importance of requirements assuming $level^3$ = 0.15. This results in $\Re^2 = \{ r_j \in \Re: \chi(r_j) \geq level^3\} = \{r_1, r_2, r_3, r_4, r_5, r_6, r_9, r_{11}, r_{12}\}$. In this case, the resulting set of requirements $\Re_3$ is estimated to exceed the given capacity bound $ef(\Re^3)$= 4886 < 4300=EFFORT. We systematically search for all those requirements, if eliminated from $\Re^3$, would result in a new subset satisfying the effort constraint. As a result of this procedure, we obtain non-dominated solutions $\Re^{3\setminus r9}$ with the property that improvement in one dimension (less effort or higher business value) can only be achieved by worsening the other. The trade-off curve between estimated effort

and associated business value is shown in Figure 4. Two solutions $\Re^{3\backslash r9}$ and $\Re^{3\backslash r5}$ are offered in correspondence to the two shaded columns in Table 2. In general, decision support is considered to offer a small set of candidate non-dominated solutions that are used to choose the final one taking into account all implicit context factors.

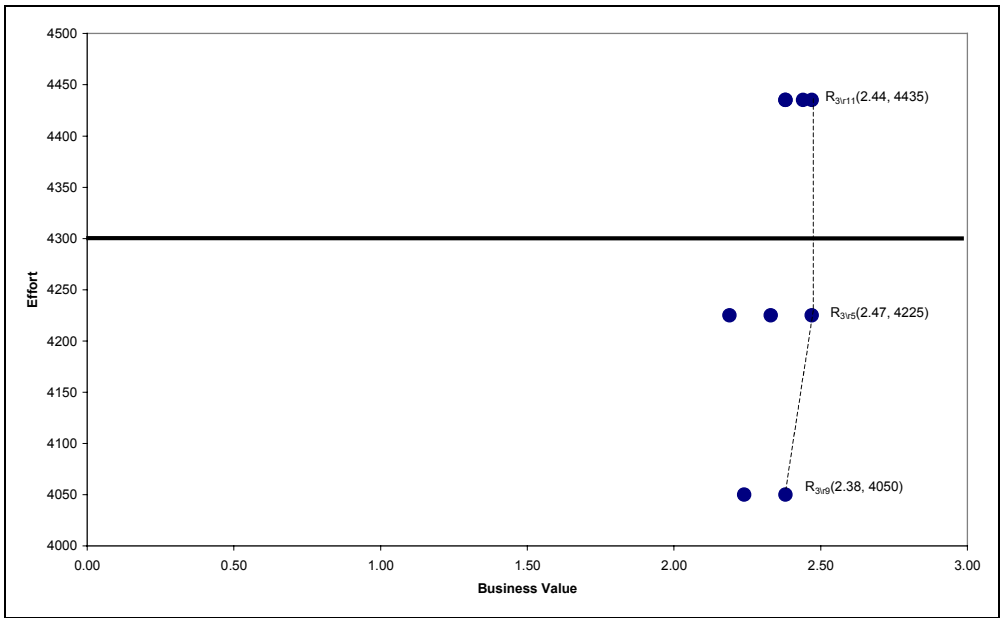

Figure 4: Trade-off curve.

7 Summary and Conclusions

One of the limitations of the Easy WinWin model is that negotiation is based on subjective measures. Which alternative will be chosen is a decision to be made by the project manager based on more or less accurate estimates. What is missing is a sound, quantitative evaluation of alternatives. In this paper, we have described a new and promising approach to support decision-making in the context of requirements selection. The added value of the Quantitative WinWin approach is its ability to offer quantitative analysis as a backbone for actual decisions. The application of Quantitative WinWin helps in the selection of requirements that meet the key needs of the most important stakeholders while still staying within the cost constraints and the required time-to-market.

The approach has been initially validated using a small-scale example with industrial data for modelling and simulation. However, the scalability of the approach still needs to be tested using a larger set of requirements. Main risks of the overall approach are (i) the availability of a sound and sufficiently detailed model for the estimation of total effort, and (ii) the availability and cooperation of stakeholders for eliciting their preference portfolio. There is a strong dependency between the quality of these two contributions and the applicability of the approach.

The presented results are part of a larger research effort on Software Engineering decision support [18]. Future research in this area will be devoted to extend the current model in terms of more sophisticated (time and effort dependent) effort constraints. In addition to that, other constraints than effort such as reliability or maintainability could be included. Finally, the full power of the underlying simulation capabilities could be used to study even more solution alternatives by varying the input parameters described in Table 1 and by defining additional parameters.

8 Acknowledgements

The authors would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC), the Alberta Software Engineering Research Consortium (ASERC) as well as the Alberta Informatics Circle of Research Excellence (iCORE) for their financial support of this research. Many thanks also to Tatyana Krivobokova for improving the GENSIM model and Trevor Croft for conducting numerical analysis using AHP.

9 References

[1] Albrecht A.J.: Measuring Application Development Productivity. Proceedings of the joint SHARE, GUIDE, and IBM Application Development Symposium pp. 83-92, 1979.

[2] Boehm B.W., Grünbacher P., Briggs B.: Developing Groupware for Requirements Negotiation: Lessons Learned, IEEE Software, May/June 2001, pp. 46-55.

[3] Boehm B.W.: A Spiral Model of Software Development and Enhancement, IEEE Computer, 21 (5), pp. 61-72, 1988.

[4] Briand L. C., Wieczorek I: Resource Estimation in Software Engineering, Fraunhofer Institute for Experimental Software Engineering, Germany, 2000, ISERN-00-05.

[5] Brooks F.P.: No Silver Bullet: Essence and Accidents of Software Engineering, IEEE Computer, 20 (4), pp. 10-19, 1987.

[6] Damian D., Eberlein A., Shaw M., Gaines B.: Using different communication media in requirements negotiation, IEEE Software 17, (3), pp. 28-35, 2000.

[7] Eberlein A.: Requirements Acquisition and Specification for Telecommunication Services, PhD Thesis, University of Wales, Swansea, UK, 1997.

[8] Forrester J.W.: Industrial Dynamics, Productivity Press, Cambridge, 1961.

[9] Lempp, P., Rudolf, L.: What Productivity Increases to Expect from a CASE Environment: Results of a User, Survey, Computer Aided Software Engineering (CASE), Ed. Chikofsky, IEEE, pp. 147-153, 1993.

[10] Paulk M.C., Curtis B., Chrissis M.B., Weber C.V.: Capability Maturity Model for Software Version 1.1, Software Engineering Institute, Technical Report CMU/SEI-93-TR24, 1993.

[11] Pfahl D., Ruhe G.: System Dynamics as an Enabling Technology for Learning in Software Organizations. In: 13th International Conference on Software Engineering and Knowledge Engineering. SEKE'2001 Skokie: Knowledge Systems Institute, 2001, pp. 355-362.

[12] Pfahl D.: An Integrated Approach to Simulation-Based Learning in Support of Strategic and Project Management in Software Organisation. Ph.D. thesis, University of Kaiserslautern, Department of Computer Science, October 2001.

[13] Reubenstein H.B. and Waters R.C.: The Requirements Apprentice: Automated Assistance for Requirements Acquisition, IEEE Transactions on Software Engineering, 17, (3), pp. 226-240, 1991.

[14] Robinson W.N., Pawlowski S.D., Volkov V: Requirements Interaction Management, Georgia State University, GSU CIS Working Paper 99-7, August 30, 1999.

[15] Saaty T.L.: The Analytic Hierarchy Process, Wiley, New York, 1980.
[16] Senge P.M.: The Fifth Discipline – the Art & Practice of the Learning Organization, Doubleday, New York, 1990.
[17] www.expertchoice.com
[18] www.seng-decisionsupport.ucalgary.ca

Document Information

Title:	Quantitative WinWin – A New Method for Decision Support in Requirements Negotiation
Date:	April 5, 2002
Report:	IESE-014.02/E
Status:	Final
Distribution:	Public