International Journal of Software Engineering
and Knowledge Engineering
Vol. 13, No. 4 (2003) 345–366

World Scientific
www.worldscientific.com

TRADE-OFF ANALYSIS FOR REQUIREMENTS SELECTION

GÜNTHER RUHE* and ARMIN EBERLEIN†

Software Engineering Decision Support Laboratory,
University of Calgary, 2500 University Drive NW, Calgary, AB T2N 1N4, Canada
**ruhe@ucalgary.ca*
†*eberlein@enel.ucalgary.ca*

DIETMAR PFAHL

Fraunhofer Institute for Experimental Software Engineering,
Sauerwiesen 6, D-67661 Kaiserslautern, Germany
pfahl@iese.fhg.de

Evaluation, prioritization and selection of candidate requirements are of tremendous importance and impact for subsequent software development. Effort, time as well as quality constraints have to be taken into account. Typically, different stakeholders have conflicting priorities and the requirements of all these stakeholders have to be balanced in an appropriate way to ensure maximum value of the final set of requirements. Trade-off analysis is needed to proactively explore the impact of certain decisions in terms of all the criteria and constraints.

The proposed method called Quantitative WinWin uses an evolutionary approach to provide support for requirements negotiations. The novelty of the presented idea is four-fold. Firstly, it iteratively uses the Analytical Hierarchy Process (AHP) for a stepwise analysis with the aim to balance the stakeholders' preferences related to different classes of requirements. Secondly, requirements selection is based on predicting and rebalancing its impact on effort, time and quality. Both prediction and rebalancing uses the simulation model prototype GENSIM. Thirdly, alternative solution sets offered for decision-making are developed incrementally based on thresholds for the degree of importance of requirements and heuristics to find a best fit to constraints. Finally, trade-off analysis is used to determine non-dominated extensions of the maximum value that is achievable under resource and quality constraints. As a main result, quantitative WinWin proposes a small number of possible sets of requirements from which the actual decision-maker can finally select the most appropriate solution.

Keywords: Requirements selection; decision support; trade-off analysis; resource constraints; Analytical Hierarchy Process (AHP); simulation.

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

*Corresponding author.

objectives and constraints. 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 the selected set of requirements and the effort required for its implementation. The overall effort that can be spent on the project is limited, the final product has to fulfill certain minimum quality constraints, 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, resource constraints and preferences of the critical stakeholders are fulfilled and the overall business value of the product is maximized.

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 their Easy 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.* [6] particularly investigate the effect of the communication media (audio, video, etc.) on the negotiation outcome.

Domain-specific support is difficult to provide. The requirements engineering community has developed some research tools [7, 12] 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.

Decision makers need support to describe, evaluate, sort, rank, select or reject candidate products, processes or tools. 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. A description and evaluation of decisions in requirements engineering is proposed by [1]. An overview of software engineering decision support is presented in [15].

Trade-off analysis aims at making the competing character of the different objectives and constraints more transparent. It is important to know to what extent the improvement in one direction (e.g., reduction of effort or duration of the project) has an impact on other criteria (e.g., quality or overall business value of the final product). More transparency regarding these relationships obviously supports the

negotiation for the most appropriate compromise solution in the context of a concrete project. However, no other approaches for requirements negotiation have yet considered trade-off analysis in a quantitative manner.

In this paper, a method for trade-off analysis called Quantitative WinWin is studied. Compared to the original method introduced in [14], the improved version places more emphasis on the consideration of effort as well as quality and time constraints. In addition to that, resource rebalancing and optimization heuristics are applied to generate a set of alternative solutions to be analyzed by a human decision-maker. Like Boehm's original WinWin [4], Quantitative WinWin uses an iterative approach, with the aim to increase and actively exploit knowledge about the requirements in all iterations. However, the difference to Boehm's WinWin groupware-based negotiation support [2] is the inclusion of quantitative methods and the emphasis on trade-off relationships in our approach. Furthermore, the evolutionary prioritization of stakeholder interests using Analytical Hierarchy Process (AHP) allows adaptation to changing environments.

Quantitative WinWin supports determination of 'best' requirements by using quantitative models in conjunction with an iterative approach that adapts to changing sets of requirements and evolving problem parameters. Emphasis is on value-based selection of requirements and resolution of conflicts in the presence of quality, time, and effort constraints. Even for medium-sized problems of a few hundred requirements, this is a problem of tremendous complexity. This is especially true in situations in which large variations in stakeholder priorities have to be considered in the requirements selection process. As a final result, Quantitative WinWin generates a small set of most promising candidate solutions from which the decision-maker can select the one which is the best match with additional implicit and subjective preferences. Overall, the process support provided by Quantitative WinWin is expected to improve transparency, effectiveness, and efficiency of decision-making.

This 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 AHP to assign priorities to stakeholders to determine priorities of the different requirements classes, and to even allow prioritization within a fixed class of requirements. Section 4 presents the simulation model GENSIM that is used in Sec. 5 for describing the overall method Quantitative WinWin. In Sec. 6, a case study describes an example with three stakeholders and three classes of requirements is described. Finally, the summary and conclusions are given in Sec. 7.

2. Underlying Assumptions and Problem Statement

2.1. *Stakeholders*

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 stakeholders. The p stakeholders are denoted as $S_1, S_2, \ldots, S_p$. Later we will use AHP to elicit both the stakeholders' preferences regarding the different classes of requirements and stakeholders preference between the individual requirements contained in the same class. As part of our evolutionary approach, these preferences can change over time.

Example stakeholders are novice, advanced or expert users as used in the example described in Sec. 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. *Classes of requirements*

The evolutionary calculation of most promising subsets of requirements under resource constraints and the prioritization of requirements from the perspective of different stakeholders is a very complex task. For even medium-sized problems of hundred requirements, the number of comparisons between requirements as the means to determine a ranking becomes prohibitively large. To address this problem, we reduce the problem size by introducing classes of requirements.

For the purpose of this paper we assume a set $\mathfrak{R} = \{r_1, \ldots, r_n\}$ of requirements $r_1, \ldots, r_n$. The set (is subdivided into q disjoint subclasses $\mathfrak{R}_1, \mathfrak{R}_2, \ldots, \mathfrak{R}_q (q < n)$ of requirements, e.g., $\mathfrak{R} = \mathfrak{R}_1 \oplus \mathfrak{R}_2 \oplus \ldots \oplus \mathfrak{R}_q$ The rationale 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 [13]) and their purpose. We assume that each class only 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. If there is a requirement that could be assigned to more than one class, we further decompose the original set $\mathfrak{R}$ by adding one more class with requirements of exactly that type. Mapping $\psi(i)$ assigns each requirement $r_i \in \mathfrak{R}$ to the associated class $\mathfrak{R}_{\psi(i)}$.

Another important aspect is the potential value of requirements. The paradigm of value-based software engineering [3] suggests to differentiate the value of individual artifacts. In our case, we assign the importance to requirements in a hierarchical way. Firstly, each class $\mathfrak{R}_k$ of requirements is assigned a value $\beta(\mathfrak{R}_k)$ that is the weighted relative priority of all stakeholders involved. Secondly, each individual requirement r_i is given a relative importance $\alpha(r_i)$ within the class $\mathfrak{R}_{\psi(i)}$. $\alpha(r_i)$ is determined again as the weighted relative priority as determined by the stakeholders. Some of the given requirements are mandatory and related to the core functionality of the system. Others may be optional. We assume that each requirement r_i has a relative importance $\alpha(r_i) \in (0, 1]$ within class $\mathfrak{R}_{\psi(i)}$ defined on a ratio scale. $\alpha(r_i) = 1$ means that the requirement is mandatory. An individual require-

ment is considered to be mandatory if it is considered to be mandatory by at least $\lceil p/2 \rceil$ of the p stakeholders.

2.3. *Constraints*

The original set of requirements $\mathfrak{R}$ is 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 a given effort bound called EFFORT. In a similar way, we define the bounds DURATION and QUALITY as the overall duration and the predicted quality of the final product, respectively. Definition of those bounds is derived from project planning. In this paper, quality is defined as the number of defects per thousand source lines of code. For this reason, we assume an upper defect bound which has to be satisfied.

We assign to each subset $\mathfrak{R}^* \subset \mathfrak{R}$ estimated values effort($\mathfrak{R}^*$), duration($\mathfrak{R}^*$) and quality($\mathfrak{R}^*$) to predict the actual impact on effort, duration, and quality, respectively. Typically, these estimates are difficult to get and uncertain in their nature. To illustrate the generic solution approach presented in Sec. 4, we assume a simulation model that is able to provide these estimates as a function of the number and average complexity of the requirements in $\mathfrak{R}^*$.

2.4. *Problem statement*

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. The application of AHP results in a normalized vector of importance $\beta = (\beta(\mathfrak{R}_1), \beta(\mathfrak{R}_2), \ldots, \beta(\mathfrak{R}_q))$ of the q classes of requirements. Applying AHP again, we get normalized values $\alpha(r_i)$ that express the weighted relative importance within a class. The absolute importance $\chi(r_i)$ of each requirement r_i is then the product of the importance of the associated class $\beta(\mathfrak{R}_{\psi(i)})$ with the relative importance $\alpha(r_i)$ of requirement r_i within class $\mathfrak{R}_{\psi(i)}$ i.e., $\chi(r_i) = \alpha(r_i) * \beta(\mathfrak{R}_{\psi(i)})$. With the notation introduced above we are now able to describe the problem "Trade-off Analysis for Requirements Selection TARS[$\mathfrak{R}$]" in a more formal way:

Trade-off Analysis for Requirements Selection TARS [$\mathfrak{R}$]

Given a set of requirements $\mathfrak{R}$, problem TARS [$\mathfrak{R}$] consists of two parts A and B. In part A, we are looking for value-maximal subsets of requirements that fulfill constraints (A1) to (A4). The final solution set Γ is determined from the solution of parts A and B. Based on the optimal value that is achievable when all constraints are fulfilled, part B aims at finding all non-dominated extensions for that value. This gives the decision-maker a comprehensive overview of what can be achieved in case of relaxation of the various constraints.

(A) Find subsets of requirements $\mathfrak{R}^* \subset \mathfrak{R}$ such that
(A1) $\{r_j \in \mathfrak{R}\text{: } \alpha(r_j) = 1\} \subset \mathfrak{R}^*$
(A2) effort($\mathfrak{R}^*$) $\leq$ EFFORT,
(A3) duration($\mathfrak{R}^*$) $\leq$ DURATION,
(A4) quality($\mathfrak{R}^*$) $\leq$ QUALITY, and
(A5) value($\mathfrak{R}^*$) $:= \sum_{r \in \mathfrak{R}^*} \alpha(r_i) * \beta(\mathfrak{R}_{\psi(i)})$ is maximum and

(B) Find subsets of requirements $\mathfrak{R}' \subset \mathfrak{R}$ such that
(B1) $\{r_j \in \mathfrak{R}\text{: } \alpha(r_j) = 1\} \subset \mathfrak{R}'$
(B2) value($\mathfrak{R}^*$) $<$ value($\mathfrak{R}'$), and
(B3) $\mathfrak{R}'$ is a non-dominated extension for value($\mathfrak{R}^*$), i.e., there is no $\mathfrak{R}^\wedge$ with
(B4) value($\mathfrak{R}'$) $<$ value($\mathfrak{R}^\wedge$),
(B5) (effort($\mathfrak{R}'$), duration($\mathfrak{R}'$), quality($\mathfrak{R}'$)) $\geq$ (effort($\mathfrak{R}^\wedge$), duration($\mathfrak{R}^\wedge$), quality($\mathfrak{R}^\wedge$))
(B6) (effort($\mathfrak{R}'$), duration($\mathfrak{R}'$), quality($\mathfrak{R}'$)) $\neq$ (effort($\mathfrak{R}^\wedge$), duration($\mathfrak{R}^\wedge$), quality($\mathfrak{R}^\wedge$))

In the following, we will describe the methodological prerequisites to solve TARS[$\mathfrak{R}$].

3. A Quantitative Hybrid Approach to Trade-off Analysis for Requirements Selection

The overall method for solving TARS consists of three main components that are described in more detail in the subsequent parts of this section:

- Evolutionary requirements prioritization using Analytic Hierarchy Process (AHP)
- Selection of candidate requirements using stepwise relaxation
- Trade-off analysis for requirements selection under resource and quality constraints

3.1. *Evolutionary requirements elicitation using the Analytic Hierarchy Process (AHP)*

Preferences or relative importance of competing alternatives are often not explicitly known. However, in order to better select requirements considering stakeholder priorities, those vectors of relative importance are needed. AHP [16] is a systematic approach to elicit preferences between different attributes. The main assumptions behind AHP are that the problem under investigation can be structured as an attributive hierarchy and that alternatives can be compared using preference ratios (for actions) or importance ratios (for criteria) from a nine-point scale. Commercial tools are available that compute the eigen-values and check the degree of consistency between the pair-wise comparisons. For the computations of the example in Sec. 6, we used the tool ExpertChoice [18].

Our problem TARS satisfies the underlying assumptions for applying AHP: Stakeholders perform pair-wise comparisons of attributes assessing their contributions to each of the higher level nodes to which they are linked. To keep the number of comparisons reasonable, we decompose the problem into two kinds of hierarchies. As both the number of classes and the number of requirements per class are assumed to be relatively small, the number of comparisons is acceptable. This number would become prohibitively large without introducing a hierarchy of requirements.

The first level of the hierarchy asks for the relative importance of the classes of requirements from the global perspective to achieve 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 as assumed. The two preference schemata are combined to rank the relative importance of the different classes of requirements for the final business value of the software product.

At the second level, the relative importance of individual requirements within its class is considered. This is done for each class of requirements. In that case, the nodes of the third level of the respective AHP graph correspond to all individual requirements of that class. Finally, the two preference schemata are combined to rank the relative importance of the individual requirements to contribute to the final business value of the software product.

3.2. *Determination of candidate requirements using stepwise relaxation*

An evolutionary approach for incremental refinement of requirements is chosen to reflect the initial impreciseness and uncertainty of requirements. The motivation for doing this 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 [3], but is valid to some extent 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 r_k with largest value $\chi(r_k)$. This is a "greedy-like" heuristic strategy suggesting the use of the most promising elements first. The assumption is that these requirements will most likely belong to the final solution set. Later, conditions are gradually relaxed to include as many requirements with lower degree of importance as possible. This is done until one of the constraints (A2) to (A4) is violated for the first time.

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 evolutionary approach where additional requirements can be included at a later stage of development.

The idea is to consider a sequence of problems TARS$[\mathfrak{R}^i]$ $i = 1, \ldots, s$. Among them, only the final one is explicitly solved. The various problems are characterized

by different sets of requirements $\mathfrak{R}^i, i = 1, \ldots, s$. Assume there is a monotonously decreasing sequence $\{\text{level}\}_{i=1,\ldots,s}$ of levels of minimum importance χ. The set $\mathfrak{R}^i$ is defined as the set of all mandatory requirements plus all the requirements having a level of importance of at least level^i, i.e., $\mathfrak{R}^i = \{r_j \in \mathfrak{R}\text{: } \alpha(r_j) = 1 \text{ or } \chi(r_j) \geq \text{level}^i\}$. This implies that $\mathfrak{R}^1 \subseteq \mathfrak{R}^2 \subseteq \ldots \subseteq \mathfrak{R}$. The number τ of iterations is defined as the first iteration at which at least one of the three constraints (A2) to (A4) is violated. In this case, the approach TARS[$\mathfrak{R}^\tau$] described in Sec. 3.3 is applied.

3.3. *Trade-off analysis for requirements selection under resource and quality constraints*

We now assume that $\mathfrak{R}^\tau = \{r_j \in \mathfrak{R}\text{: } \alpha(r_j) = 1 \text{ or } \chi(r_j) \geq \text{level}^\tau\}$ is the set of requirements determined by iteration τ when some of the additional constraints (A2) to (A4) are initially violated. A heuristic approach is applied to solve TARS[$\mathfrak{R}^\tau$]. The approach uses two basic operations to compute a set of non-dominated extensions fulfilling (B1) to (B6). These operations are called REBALANCE and MODIFY.

REBALANCE tries to remove constraint violations by gradually relaxing other criteria, hoping to stay within acceptable limits for all defined constraints. The rationale for this step is that there is a trade-off between effort, duration and quality. Improving one of these criteria will typically deteriorate the others (assuming no technology change).

In general, the elimination of all constraint violations cannot be achieved by solely applying REBALANCE. In cases where REBALANCE fails, operation MODIFY uses reductions and extensions systematically to the requirements set under investigation. Assume we created a set of requirements $\mathfrak{R}^*$ which cannot be implemented within the given constraints, even after the application of the REBALANCE step. To achieve feasibility again, one has to eliminate some requirement(s) from $\mathfrak{R}^*$. Applying rule M1 as specified below tries to eliminate constraint violations by removing the requirement(s) with the minimum contribution to the overall business value. However, after this elimination, it might be possible to add again another requirement, i.e., rule M2 should be applied as specified below.

Rule M1:

(1) Delete requirement r_k from the set of requirements $\mathfrak{R}^*$ with $\chi(r_k) = \min\{\chi(r_j) | r_j \in \mathfrak{R}^* \text{ and } \alpha(r_j) < 1\}$.
(2) In case there is more than one requirement with the same minimal value, then take the one r_j with the maximal complexity ($\text{compl}(r_j) = \max\{\text{compl}(r_j) | r_j \in \mathfrak{R}^* \text{ and } \chi(r_i) = \chi(r_k)\}$).

Rule M2:

(1) Add requirement r_k to the set of requirements $\mathfrak{R}^*$ with $\text{compl}(r_k) = \min\{\text{compl}(r_j) | r_j \notin \mathfrak{R}^*\}$.

(2) In case there is more than one requirement with the same minimal complexity, then take the one r_j with the maximal value ($\chi(r_j) = \max\{\chi(r_j) | r_j \notin \mathfrak{R}^*$ and $\text{compl}(r_i) = \text{compl}(r_k)\}$).

4. Project Estimation Using GENSIM

Even though inherently difficult, we assume that an estimation method that considers effort, duration, and quality can be applied based on a given specification of requirements. As discussed in [5], estimation is increasingly based on a combined use of expert opinion and simulation. In order to generate estimates effort($\mathfrak{R}$), duration($\mathfrak{R}$), and quality($\mathfrak{R}$) for a given set of requirements and a fixed average staffing we used the simulation model GENSIM (GENeric SIMulator) [11]. The GENSIM model simulates the software development process from the end of the requirement analysis step through to the end of system testing (for further details see [11] and [14]). 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 behavior of each development cycle in Boehm's spiral model. For producing the effort estimates used in the example in Sec. 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 the following five interrelated sub-models:

- **Production:** This sub-model represents a typical software development cycle consisting of the following steps of transitions (cf. Fig. 1): 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 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.

The most important input and output parameters and their use in the context of predicting effort, quality and duration are listed in Table 1. The

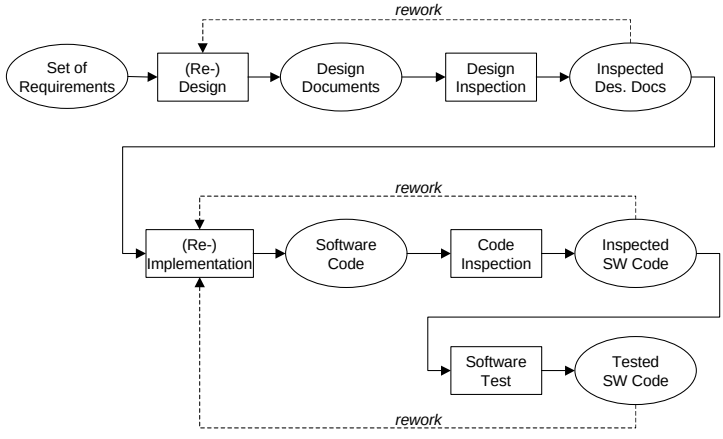

Fig. 1. Schematic representation of the product flow captured by the GENSIM production submodel.

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

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) [days]
Planned_completion_time (optional) [days]	Effort (project total and per phase) [person days]
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]	

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 Sec. 6, all gray-shaded input parameter have been varied as part of the simulation runs.

The simulation modeling approach used to develop GENSIM has been defined in [11] 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 [17]. 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. The schematic conventions of flow graphs are shown in Fig. 2.

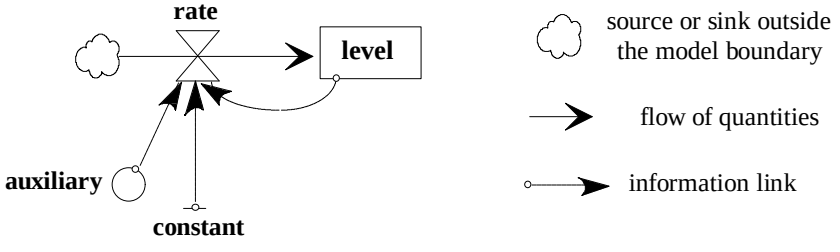

Fig. 2. Schematic conventions of flow graphs.

With the schematic conventions shown in Fig. 2, the flow graph representation of the GENSIM product flow (cf. Fig. 1) can be represented as shown in Fig. 3.

Even though System Dynamics based simulation modeling 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:

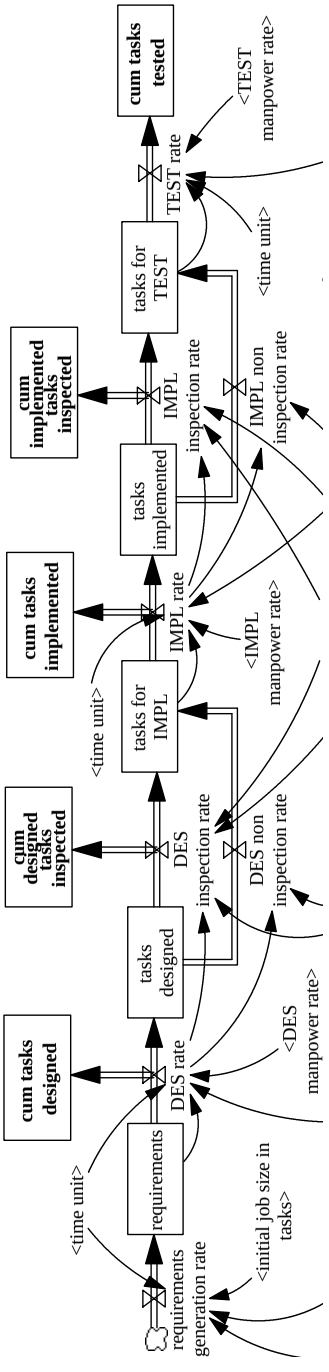

Fig. 3. Flow graph of GENSIM's production sub-model (extract).

- 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 within the organization, e.g., CMM level 3 or higher [9].
- 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 as 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 Secs. 3 and 4. Quantitative WinWin consists of three phases called Initialization (Phase 1), Iteration (Phase 2), and Termination (Phase 3). In the following, we describe the three phases in more detail.

5.1. *Phase 1: Initialization*

At the beginning we define our initial set of requirements by only looking at mandatory requirements, i.e., $\mathfrak{R}^0 = \{r_j \in \mathfrak{R}\colon \alpha(r_j) = 1\}$. We check feasibility of $\mathfrak{R}^0$. If one of the constraints (1) to (3) is violated for $\mathfrak{R}^0$ then we apply REBALANCE as described in Sec. 3.3. If no feasibility can be achieved this way, the problem does not have a feasible solution.

5.2. *Phase 2: Iteration*

During each iteration, six consecutive steps are applied as described below and illustrated in Fig. 4. The number of iterations is not determined in advance. It depends on the degree of change in requirements during the development cycle and the sequence of thresholds defined by the expert.

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 that have an

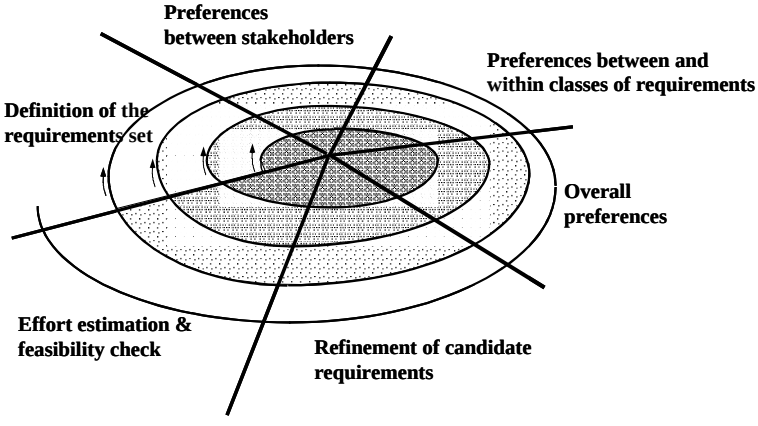

Fig. 4. Principal steps of Quantitative WinWin.

importance value of at least level^i, i.e., $\mathfrak{R}^i = \{r_j \in \mathfrak{R}\text{: } \alpha(r_j) = 1 \text{ or } \chi(r_j) \geq \text{level}^i\}$. 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. Another possible modification of the candidate set of requirements is that further requirements can be added to $\mathfrak{R}$ at later iterations (as can be seen in the example in Sec. 6).

Step 2: Computation of preferences between involved stakeholders

Preferences between involved stakeholders are computed from the perspective of the overall business value. AHP is applied for that purpose resulting in a normalized vector of weights weight^0 = (weight$_{0,1}$, ..., weight$_{0,p}$) with Σ weight$_{0,j} = 1$.

Step 3: Computation of preferences between and within requirements classes

Preferences between requirements classes are computed from the perspective of the p individual stakeholders. AHP is applied for that purpose resulting in normalized vectors weight_c^1, ..., weight_c^p. Each of them is of dimension q.

Analogously, preferences between individual requirements of all (fixed) class are computed from the perspective of the p individual stakeholders. AHP is applied for that purpose resulting in normalized vectors weight_r^1, ..., weight_r^p

Step 4: Computation of overall preferences between and within requirements classes

Computation of overall preferences between requirements classes by consecutive application of the weights computed in Step 3 (the vectors weight_c^1, ..., weight_c^p

are arranged as the columns 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 $\beta = (\beta(\mathfrak{R}_1), \beta(\mathfrak{R}_2), \ldots, \beta(\mathfrak{R}_q))$ of the q classes of requirements.

Analogously, computation of overall preference of individual requirements of all (fixed) class by consecutive application of the weights computed in Step 3 (the vectors weight_$r^1, \ldots,$ weight_r^p are arranged as the columns 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 $\alpha = \alpha(r_1), \alpha(r_2), \ldots, \alpha(r_n))$ of the n individual requirements. An individual requirement is considered to be mandatory if it is considered to be mandatory by at least $\lceil p/2 \rceil$ of the p stakeholders.

Step 5: Refinement of candidate requirements

Subset $\mathfrak{R}^i = \{r_j \in \mathfrak{R}\text{: } \alpha(r_j) = 1 \text{ or } \chi(r_j) \geq \text{level}^i\}$ with $\chi(r_i) = \alpha(r_i) * \beta(\mathfrak{R}_{\psi(i)})$ is defined where $\mathfrak{R} = \mathfrak{R} + \Delta\mathfrak{R}^i$ is the original set of requirements eventually extended by requirements $\Delta\mathfrak{R}^i$ added at a later stage.

Step 6: Feasibility check

At each iteration I, an estimation is done to compute effort($\mathfrak{R}^i$), quality($\mathfrak{R}^i$), and duration($\mathfrak{R}^i$) for the actual set $\mathfrak{R}^I$ of requirements under investigation. The set of requirements is feasible if and only if effort($\mathfrak{R}^i$) $\leq$ EFFORT, quality($\mathfrak{R}^i$) $\leq$ QUALITY, duration($\mathfrak{R}^i$) $\leq$ DURATION. If the set is not feasible then we go to Phase 3.

5.3. *Phase 3: Termination*

We now assume that $\mathfrak{R}^\tau = \{r_j \in \mathfrak{R}\text{: } \alpha(r_j) = 1 \text{ or } \chi(r_j) \geq \text{level}^\tau\}$ is the set of requirements determined by iteration τ when some of the additional resource constraints (A2) to (A4) are initially violated. A heuristic approach called TARS[$\mathfrak{R}^\tau$] is applied for trade-off analysis. It uses REBALANCE and iteratively applies Rules M1 and M2 of MODIFY as described in Sec. 3.3.

After each modification of the set of requirements it is necessary to check whether any of the project constraints are violated. If this is the case, REBALANCE will be applied in order to find a parameter setting that fulfils the constraints. The heuristic approach is using interaction with the decision-maker to finally decide when to stop computing the set Γ of alternative solutions. The default termination is that all solutions generated from the application of REBALANCE and MODIFY are dominated by existing ones.

6. Initial Experience Using the Approach — An Example

We report initial experiences with the proposed approach by considering the example of the development of a software product that has three classes of potential 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 consider different

Table 2. Basic information about the sequence of iterations and the result of re-balancing.

Subset characteristics	Class r_j	$\mathfrak{R}_{\psi(j)}$	$\alpha(r_j)$	$\beta(\mathfrak{R}_{\psi(j)})$	$\chi(r_j)$	$\mathfrak{R}^0$	$\mathfrak{R}^1$	$\mathfrak{R}^2$	$\mathfrak{R}^3$	$\mathfrak{R}^{3'}$
	r_1	$\mathfrak{R}_1$	0.6	0.404	0.24			X	X	X
	r_2	$\mathfrak{R}_1$	0.9	0.404	0.36		X	X	X	X
	r_3	$\mathfrak{R}_1$	**1.0**	0.404	0.40	X	X	X	X	X
	r_4	$\mathfrak{R}_1$	0.7	0.404	0.28		X	X	X	X
	r_5	$\mathfrak{R}_2$	**1.0**	0.354	0.35	X	X	X	X	X
	r_6	$\mathfrak{R}_2$	0.6	0.354	0.21				X	
	r_7	$\mathfrak{R}_2$	**1.0**	0.354	0.35	X	X	X	X	X
	r_8	$\mathfrak{R}_2$	0.5	0.354	0.18					X
	r_9	$\mathfrak{R}_3$	**1.0**	0.268	0.27	X	X	X	X	X
	r_{10}	$\mathfrak{R}_3$	0.3	0.268	0.08					
	r_{11}	$\mathfrak{R}_3$	0.8	0.268	0.21	n/a	n/a		X	
	r_{12}	$\mathfrak{R}_3$	**1.0**	0.268	0.27	n/a	n/a	n/a	X	X
level^1						n/a	0.25			
level^2								0.22		
level^3									0.20	n/a
Total size						400	600	700	1000	900
Ø complexity						3.00	3.00	2.71	2.80	2.33
effort($\mathfrak{R}^i$)						2704	4199	4706	*7241**	5431
quality($\mathfrak{R}^i$)						1.02	1.01	1.01	1.02	1.02
duration($\mathfrak{R}^i$)						325	375	402	*558**	425
$\Sigma_{r\in\mathfrak{R}^i}\chi(r)$						1.37	2.01	2.25	2.95	2.71

classes of requirements as 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 $\mathfrak{R}_1, \mathfrak{R}_2$, and $\mathfrak{R}_3$. 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 $\mathfrak{R}_1$)
- Usability (class $\mathfrak{R}_2$)
- Security and reliability (class $\mathfrak{R}_3$)

We use GENSIM as introduced in Sec. 4 to model and simulate this example. Basic information about the example and the results of three iterations are summarized in Table 2. Threshold levels for evolutionary selection of requirements are $\text{level}_1 = 0.25$, $\text{level}_2 = 0.22$ and $\text{level}_3 = 0.20$. $\alpha(r_j)$, $\beta(\mathfrak{R}_{\psi(j)})$, and $\chi(r_j)$ describe the relative importance of requirement r_j within class $\mathfrak{R}_{\psi(j)}$, importance of class $\mathfrak{R}_{\psi(j)}$, and global importance of requirement r_j, respectively. The six final rows of the table show the total size of the product (in size units), the average complexity of the implemented requirements (on a scale between 1 and 5), the estimates for effort($\mathfrak{R}^i$), quality($\mathfrak{R}^i$), and duration($\mathfrak{R}^i$) of sets $\mathfrak{R}^i$, and, eventually, the business value $\Sigma_{r\in\mathfrak{R}^i}\chi(r)$ associated with $\mathfrak{R}^i$.

The following constraints are assumed in this example: The total effort available to implement the final set of requirements is set to EFFORT = 5600 person days. The minimal acceptable quality level is set to QUALITY = 1.2 defects per size unit. The maximal acceptable duration of the project is set to DURATION = 430. The average manpower is limited to maximal 13 developers, and the development process as well as the skill levels of the manpower is assumed to be fixed. Each requirement is assumed to have a fixed size and specific complexity (low, medium or high).

We assume the product manager has to decide which of the initially given ten (later, there will be twelve) requirements will be selected to maximize the overall business value. In Phase 1, the set $\mathfrak{R}^0$ of all mandatory requirements is defined. As the set is feasible in terms of constraints (A2) to (A4), we proceed with Phase 2.

To illustrate the concept of stepwise refinement, we assume three iterations. In each iteration, the requirements acceptance threshold is relaxed. In order to simplify the example, we assume that the preference in AHP described by the four 3×3 matrices does not change over time. As stated above, Quantitative WinWin in general allows changing of priorities.

A 3×3 matrix M^0 of preferences between the three types of users in terms of the overall goal to maximize the business value is built. Matrices M^1, M^2, and M^3 describe the preferences of the three classes of requirements from the perspective of stakeholder S_1, S_2, and S_3, respectively. This may look as follows:

$$M^0 = \begin{bmatrix} 1 & 3 & 6 \\ \frac{1}{3} & 1 & 4 \\ \frac{1}{6} & \frac{1}{4} & 1 \end{bmatrix}, \; M^1 = \begin{bmatrix} 1 & 2 & 5 \\ \frac{1}{2} & 1 & 4 \\ \frac{1}{5} & \frac{1}{4} & 1 \end{bmatrix}, \; M^2 = \begin{bmatrix} 1 & \frac{1}{3} & \frac{1}{6} \\ 3 & 1 & \frac{1}{2} \\ 6 & 2 & 1 \end{bmatrix}, \; M^3 = \begin{bmatrix} 1 & \frac{1}{5} & \frac{1}{2} \\ 5 & 1 & 4 \\ 2 & \frac{1}{4} & 1 \end{bmatrix}.$$

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 stakeholders (i.e., novice, advanced, expert) for the final business value from the perspective of the product manager (Step 2).
- $\text{weight\_}c^1$ = (0.570, 0.333, 0.097) gives the importance of the three classes of requirements from the perspective of the novice user (Step 3.1).
- $\text{weight\_}c^2$ = (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).
- $\text{weight\_}c^3$ = (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 $\beta = (\beta(\mathfrak{R}_1), \beta(\mathfrak{R}_2), \beta(\mathfrak{R}_3))$ which is the vector of importance of the three classes of

requirements for the overall business value (Step 4):

$$\begin{matrix}(\mathfrak{R}_1)\\(\mathfrak{R}_2)\\(\mathfrak{R}_3)\end{matrix}\begin{bmatrix}0.570 & 0.100 & 0.117\\0.333 & 0.300 & 0.683\\0.097 & 0.600 & 0.200\end{bmatrix}\begin{bmatrix}0.644\\0.271\\0.085\end{bmatrix}\begin{matrix}(S_1)\\(S_2)\\(S_3)\end{matrix}=\begin{bmatrix}0.404\\0.354\\0.268\end{bmatrix}\begin{matrix}(\beta(\mathfrak{R}_1))\\(\beta(\mathfrak{R}_2))\\(\beta(\mathfrak{R}_3))\end{matrix}$$

According to the individual scorings, the first class of requirements is most important, and the third class is the least important. Eventually, these scores could be changed as part of the stepwise refinement approach. However, we will not consider such changes in this example. For the sake of simplicity, we further assume that the vector $\alpha = (\alpha(r_1), \alpha(r_2), \ldots, \alpha(r_n))$ as shown in Table 2 (determined in Steps 3 and 4) is the result of a weighted sum computation of stakeholder priorities.

Iteration 1: The initial requirements acceptance threshold level of importance, level^1, is set to 0.25. This results in $\mathfrak{R}^1 = \{r_j \in \mathfrak{R}\text{: } \alpha(r_j) = 1 \text{ or } \chi(r_j) \geq \text{level}^1\} = \{r_2, r_3, r_4, r_5, r_7, r_9\}$. The estimates effort$(\mathfrak{R}^1) = 4199$, quality$(\mathfrak{R}^1) = 1.01$, and duration$(\mathfrak{R}^1) = 375$ satisfy the given bounds EFFORT = 5600, QUALITY = 1.2, and DURATION = 430, respectively.

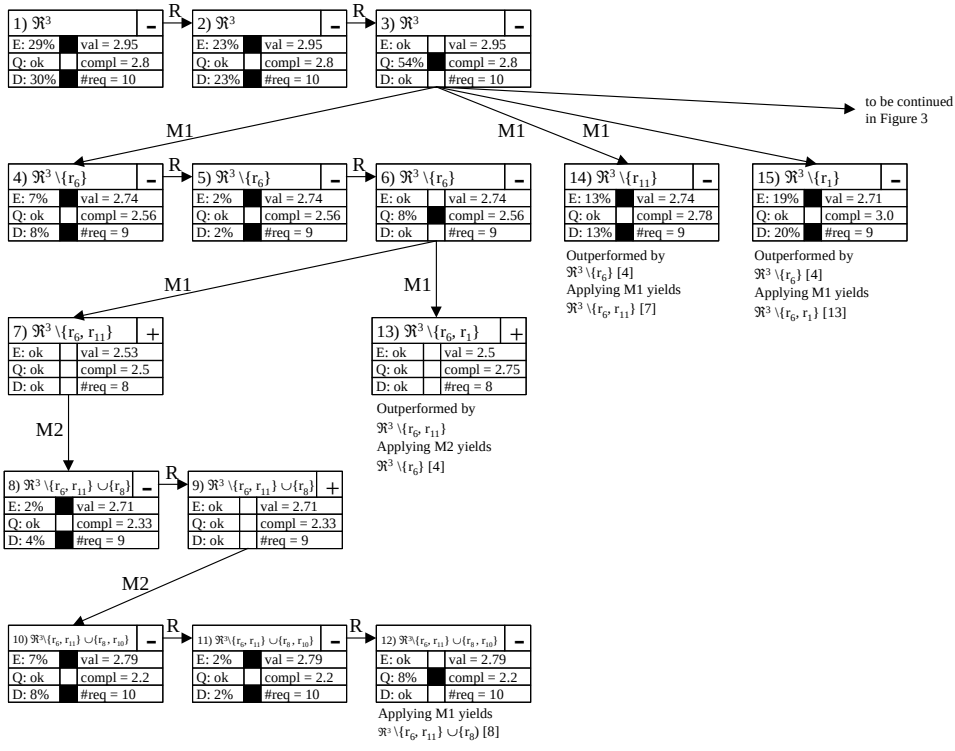

Fig. 5. Evolution of generated solutions applying operations REBALANCE (R) and MODIFY (rules M1 and M2). Each box represents exactly one solution generated. Each solution is described in terms of its relative fulfillment of the three additional constraints related to effort (E), quality (Q), and duration (D), the respective business value (val), average complexity (compl), and the number of included requirements (#req).

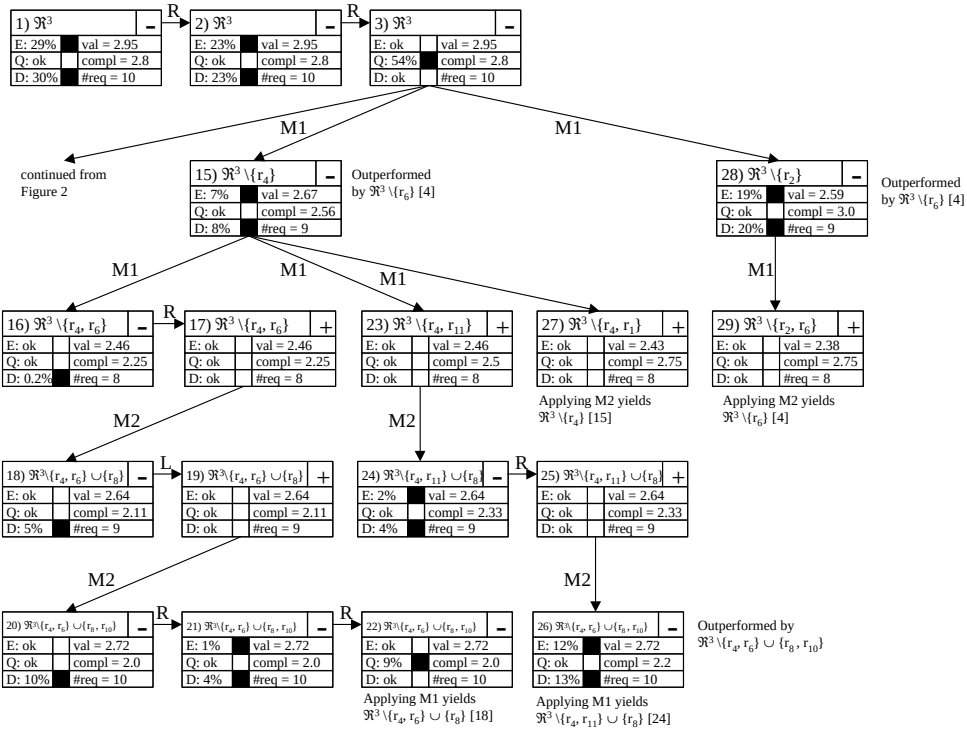

Fig. 6. Continued results of applying operations REBALANCE (R) and MODIFY (rules M1 and M2).

Iteration 2: In the second iteration, we have a relaxed level of importance to level$^2 = 0.22$. Furthermore, an additional requirement r_{11} is considered to be included into the requirements set. Following the steps of Phase 2, we get our new set $\mathfrak{R}^2 = \{r_j \in \mathfrak{R}\text{: } \alpha(r_j) = 1 \text{ or } \chi(r_j) \geq \text{level}^2\} = \{r_1, r_2, r_3, r_4, r_5, r_7, r_9\}$ which still does not violate the given constraints (effort$(\mathfrak{R}^2) = 4706 < 5600 =$ EFFORT, quality$(\mathfrak{R}^2) = 1.01 < 1.2 =$ QUALITY, duration$(\mathfrak{R}^2) = 402 < 430 =$ DURATION).

Iteration 3: In the third iteration, we again add a new requirement r_{12}. In addition to that, we further relax the required level of importance of requirements assuming level$^3 = 0.20$. This results in $\mathfrak{R}^3 = \{r_j \in \mathfrak{R} : \alpha(r_j) = 1 \text{ or } \chi(r_j) \geq \text{level}^3\} = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_9, r_{11}, r_{12}\}$. In this case, the resulting set of requirements $\mathfrak{R}^3$ is estimated to violate two constraints, i.e., effort$(\mathfrak{R}^3) = 7241 > 5600 =$ EFFORT and duration$(\mathfrak{R}^3) = 558 > 430 =$ DURATION.

Having violated feasibility constraints (A2) to (A4) for the first time, we enter Phase 3. The subsequent applications of operations REBALANCE (abbreviated by L) and MODIFY (rules A and B) results in an evolution of solutions to build a final solution set Γ. This is illustrated in Figs. 5 and 6. Each box represents exactly one solution generated. In addition to the composition of the new solution,

it gives the relative fulfillment of the three additional constraints related to effort (E), quality (Q), and duration (D). Furthermore, the respective business value, average complexity, and the number of included requirements are given for each of the generated sets.

A first attempt to resolve the constraint violation caused by $\mathfrak{R}^3$ is to relax the quality parameter (abbreviated by Q) in order to reduce effort (E) and duration (D). As can be seen in Fig. 5, however, this attempt fails for set $\mathfrak{R}^3$. Although it is possible to reduce the excess of limits set by the constraints (compare box 2 with box 1 in Fig. 5), it is not possible to satisfy all constraints at the same time (cf. boxes 2 and 3).

Since REBALANCE operations on project parameters for set $\mathfrak{R}^3$ do not yield a satisfactory solution, rules M1 and M2 have to be applied as described in Sec. 3.3. The application of rules M1 and M2 should stop when the project manager does not expect any further improvement. In the example, the decision to stop was made after all feasible reductions of set $\mathfrak{R}^3$ by one requirement had been examined. It turns out that the highest business value is achieved with set $\mathfrak{R}^{3'} = \{r_1, r_2, r_3, r_4, r_5, r_7, r_8, r_9, r_{12}\}$ (cf. box 9).

The solution set Γ with all the solutions satisfying (B1) to (B6) is given in Table 3.

Table 3. Solution set Γ for TARS[$\mathfrak{R}$].

Set #	Requirements r_i	Value( )	Fulfillment (A2)	Fulfillment (A3)	Fulfillment (A4)
2	{1,2,3,4,5,6,7,9,11,12}	2.95	−23%	yes	−23%
3	{1,2,3,4,5,6,7,9,11,12}	2.95	yes	−54%	yes
11	{1,2,3,4,5,7,8,9,10,12}	2.79	−2%	yes	−2%
12	{1,2,3,4,5,7,8,9,10,12}	2.79	yes	−8%	yes
20	{1,2,3,5,7,8,9,10,11,12}	2.72	yes	yes	−10%
21	{1,2,3,5,7,8,9,10,11,12}	2.72	−1%	yes	−4%
9	{1,2,3,4,5,7,8,9,12}	2.71	yes	yes	yes

7. Summary and Conclusions

One of the limitations of Boehm's 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. Application of Quantitative WinWin helps in the selection of requirements that meet the key needs of the most important stakeholders. It gives the best value achievable when all resource and quality constraints are met. In addition to that, it shows how the

business values can be improved as more resources become available and/or the quality constraint is relaxed.

The novelty of the presented idea is four-fold. Firstly, requirements selection is based on predicting and rebalancing its impact on effort, time and quality. Secondly, AHP is used iteratively for a stepwise analysis to balance the stakeholders' preferences related to the different classes of requirements. Both prediction and rebalancing are based on the modeling and simulation prototype GENSIM. Thirdly, alternative solution sets offered to the decision-maker are developed incrementally based on thresholds for the degree of importance of requirements and heuristics that allow us to find a best fit considering given constraints. Finally, trade-off analysis is used to determine non-dominated extensions of the maximum value that is achievable under resource and quality constraints. As a main result, quantitative WinWin proposes a small number of alternative sets of requirements from which the actual decision-maker can finally select the most appropriate one.

The approach has been initially validated using a small-scale example for modeling and simulation. However, the scalability of the approach still needs to be tested using a larger set of requirements. The main risks of the overall approach are (i) availability of a sound and sufficiently detailed model for the estimation of total effort, quality, and duration, and (ii) availability and cooperation of stakeholders for eliciting their preference portfolio. The applicability of the Quantitative WinWin approach strongly depends on the quality of these two contributions.

Using quantitative measures during the requirements engineering process is inherently difficult. This paper is an attempt to assist the developer in making trade-off decisions using simulations. In order to achieve this, our research uses simplifications of reality. We assume that requirements are independent even though many requirements depend on each other or have an impact on each other. Especially non-functional requirements tend to have numerous links to functional requirements. Such dependencies between requirements have an influence on the value and complexity of combinations of requirements. On the other hand, the complexity of implementing a large number of requirements is likely to be higher than the complexity of individual requirements since issues such as feature interaction have to be considered. It can be assumed that the value and complexity of individual requirements are not constant but increase with the number of selected requirements. This area of requirements dependence will be investigated in future.

Acknowledgements

The authors would like to qthank the Alberta Informatics Circle of Research Excellence (iCORE) Natural Sciences, the Engineering Research Council of Canada (NSERC), and the Alberta Software Engineering Research Consortium (ASERC) for their financial support of this research. Comments of anonymous referees have contributed to further improve the clarity of the paper. Many thanks also to An Ngo The for stimulating discussions on former versions of the paper and to Tatyana Krivobokova for improving the GENSIM model.

References

1. A. Aurum and C. Wohlin, "Applying decision making frameworks in requirements engineering", *8th Int. Workshop on Requirements Engineering: Foundation for Software Quality* (*REFSQ'02*), 9–10 September, Essen, Germany.
2. B. W. Boehm, P. Grünbacher and B. Briggs, "Developing groupware for requirements negotiation: Lessons learned", *IEEE Software*, May/June 2001, pp. 46–55.
3. B. W. Boehm and K. Sullivan, "Value-based software engineering", *Tutorial at 25th Int. Conf. on Software Engineering*, Portland, 2003.
4. B. W. Boehm, "A spiral model of software development and enhancement", *IEEE Computer* **21**(5) (1988) 61–72.
5. L. C. Briand and I. Wieczorek, "Resource estimation in software engineering", in *Encyclopedia of Software Engineering*, ed. J. Marcinicak, 2002, pp. 1160–1196.
6. D. Damian and A. Eberlein, M. Shaw, and B. Gaines, "Using different communication media in requirements negotiation", *IEEE Software* **17**(3) (2000) 28–35.
7. A. Eberlein, "Requirements Acquisition and Specification for Telecommunication Services", PhD Thesis, University of Wales, Swansea, UK, 1997.
8. J. W. Forrester, *Industrial Dynamics*, Productivity Press, Cambridge, 1961.
9. M. C. Paulk., B. Curtis, M. B. Chrissis and C. V. Weber, "Capability Maturity Model for Software Version 1.1", Software Engineering Institute, Technical Report CMU/SEI-93-TR24, 1993.
10. D. Pfahl and G. Ruhe, "System dynamics as an enabling technology for learning in software organizations", in *13th Int. Conf. on Software Engineering and Knowledge Engineering* (*SEKE'2001*) Skokie: Knowledge Systems Institute, 2001, S. 355–362.
11. D. Pfahl, "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.
12. H. B. Reubenstein and R. C. Waters, "The requirements apprentice: Automated assistance for requirements acquisition", *IEEE Trans. on Software Engineering* **17**(3) (1991) 226–240.
13. W. N. Robinson, S. D. Pawlowski and V. Volkov, "Requirements Interaction Management", Georgia State University, GSU CIS Working Paper 99-7, August 30, 1999.
14. G. Ruhe, A. Eberlein and Pfahl, "Quantitative WinWin — A new method for decision support in requirements negotiation", *Proc. 14th Int. Conf. on Software Engineering and Knowledge Engineering* (*SEKE'2002*), Ischia, Italy, July 2002, pp. 159–166.
15. G. Ruhe, "Software engineering decision support — A new paradigm for Learning Software Organizations", in *Proc. 4th Workshop on Learning Software Organizations*, Springer 2003.
16. T. L. Saaty, *The Analytic Hierarchy Process*, Wiley, New York, 1980.
17. P. M. Senge, *The Fifth Discipline — the Art & Practice of the Learning Organization*, Doubleday, New York, 1990.
18. www.expertchoice.com