Complementing XP with Requirements Negotiation

Paul Grünbacher **Christian Hofer**

Systems Engineering and Automation
Johannes Kepler University
Altenbergerstr. 69, 4040 Linz, Austria
+43 732 2468 {x8867 | x8873}
{pg | ch}@sea.uni-linz.ac.at

ABSTRACT
Attaining consensus among the success-critical stakeholders is crucial for the success of any software engineering project. Extreme Programming (XP) addresses this fact by providing a set of negotiation-oriented practices. In this paper we discuss negotiation techniques that would nicely complement XP. We present the EasyWinWin requirements negotiation approach and discuss its potential benefits for XP.

Keywords
Extreme Programming, Requirements Negotiation, Stakeholder Win-Win Concepts.

INTRODUCTION
In any software project regardless of using an agile or a plan-driven approach attaining consensus among the system stakeholders (e.g., customers, developers, users, or marketing experts, etc.) about the requirements and constraints is crucial. In a negotiation process stakeholders have to balance their personal goals with the preferences of others to come up with mutually satisfactory solutions and to develop a shared vision and a joint understanding about the development goals and alternatives. Adopting systematic requirements negotiation processes and techniques that support gathering, elaborating, and negotiating the stakeholder objectives can positively influence project success in several aspects [5, 11]:

More complete requirements. Stakeholders typically have to deal with incomplete and often vaguely defined requirements. Although not all requirements can be identified initially and thus have to be evolved and refined during development, effective negotiation processes can help to elaborate a more complete set of requirements in the early stages of a project.

Enabling a shared vision. Only few projects will be based on stable requirements, more typically the basic conditions change during development, which inevitably affects the requirements. In such situations a shared vision among the stakeholders becomes crucial. A negotiation approach can provide adequate guidance for developing a shared vision and support for renegotiation.

Dealing with changes. In any project some of the originally taken assumptions will become invalid at a later stage, which means that decisions already taken must be revised. Negotiation methods ease this task as they preserve the rationale of decisions taken earlier in a project.

Negotiation becomes even more important in the increasing dynamics of today's business world characterized by rapid innovation cycles leading to radical rethinking and the re-organization of software development processes. Projects often have to pursue moving targets and to check continuously whether the originally defined bundle of objectives is still valid under the changed circumstances [9]. Agile software development methods like Extreme Programming (XP) have been developed to deal with that challenge and to foster reconsideration and modification of decisions [1]. Although these methods provide some guidance for negotiation (e.g., the "Planning Game") we think that the role of negotiations should be emphasized.

The remainder of this paper is organized as follows: We will first discuss selected XP practices and show how they support negotiations. As an example of a negotiation method we will then present the EasyWinWin requirements negotiation approach. In the conclusion we discuss its potential benefits for XP.

REQUIREMENTS NEGOTIATION AND XP
The XP-approach provides a set of fundamental practices allowing the production of high-quality software in the context of a rapidly changing environment. Several of the 12 XP practices [2] are related to requirements negotiation and elicitation of user needs: The *metaphor* for example supports a shared vision among stakeholders, the *planning game* helps to understand and prioritize requirements, the idea of the *on-site customer* improves stakeholder involvement, and the specification of acceptance *tests* helps to refine und evolve requirements:

Metaphor
In the beginning of a software project the fundamental vision and mission should be determined and described in a superordinate, fundamental metaphor. Beck describes the system metaphor as a "story that everyone […] can tell about how the system works" [2]. Specific aspects of a system can be described in subordinated, but more concrete metaphors.

Finding an appropriate metaphor relies on involving all success-critical stakeholders, discussing their different views and negotiating all relevant details like the key objects of the domain and their interaction. The metaphor is a crucial 'ingredient' of the shared system vi-

sion as it helps to align all involved stakeholders towards the same goals and architectural foundations, and harmonizes the common project language [12]. Broadly speaking the metaphor is the underlying foundation for the requirements to be developed and can be subject to change during development as the requirements themselves.

Planning Game
The XP planning game puts a special focus on the negotiation of the requirements and likewise on the planning of their implementation. All members of the project team meet and discuss the requirements captured by the customer in so-called user stories.

In a negotiating process the stakeholders try to achieve consensus about the user stories and the related implementation effort: While the customer describes the functional aspects of the user stories, the developers evaluate their estimated implementation efforts and associated risks. Based on that agreement the customer selects the user stories and allocates them into the respective releases – another negotiation process. User stories with the highest expected benefits will be implemented first, in order to deliver the maximum business value to the customer in each release.

Subsequently iteration planning is executed. The user stories of a particular release will be implemented during several, shorter iterations. Each user story is divided into several tasks, which are estimated and negotiated regarding their effort by the programmers. Finally, the programmers take responsibility for the tasks and start to implement them. The customer is present during iteration planning and helps the programmers, if they get stuck with the estimation [3].

On-site customer
XP emphasizes the involvement of the customer in the development process. The on-site customer is responsible for supplying user stories in a form assessable for the programmers. User stories can describe both functional and non-functional characteristics of the system. When needed the customer gives immediate help and feedback throughout the project. Furthermore the on-site customer is responsible for defining and checking the acceptance tests. It is important to note that the on-site customer should not be understood as only one physical person, but as a representative for all stakeholders involved from the customer side [4].

Testing
For each user story at least one acceptance test must be provided by the on-site customer to check the fulfillment of both functional and non-functional requirements. The specification of acceptance tests supports the programmers to understand the customer's requirements in more detail and helps the on-site customer to re-check the completeness, consistency, and reliability of the requirements. The specification of acceptance tests leads to a re-interpretation and refinement of the requirements defined earlier in the development process, potentially leading to a re-negotiation.

If all acceptance tests existing for a user story are passed, the team can be sure that the required functionality has been implemented as requested. Since both the metaphor and the requirements themselves can be modified, passed acceptance tests must be re-executed even in later iterations and releases regularly to determine contingent modifications of the requirements or unintentional interactions caused by the modification. In addition it is desirable, that – like unit tests – the acceptance tests will be executed automatically [10].

Negotiation-related Issues
XP practices support the identification, negotiation, and evolvement of requirements as discussed above. There are, however, some potential problems and risks:

Insufficient definition of the project vision/mission. The definition of high-level goals and constraints (e.g., cost, schedule, staffing, interfaces, technology, etc.) in a project vision/mission is not an explicit part of XP, which may have some negative implications. For example, without a project mission it is difficult to define a consistent set of user stories in the planning game.

Limited set of stakeholders. Success-critical stakeholders are not always limited to customers and developers. A mechanism to identify and involve all success-critical stakeholders in the planning game would be beneficial. This might for example include end-users, marketing experts, financial experts, executives, or even the general public in some cases. Stakeholders need to be represented and integrated in the definition process for the metaphor as well as in the negotiation process for the requirements. Neglecting important stakeholders can lead to incomplete requirements and to a wrong set of objectives, which can lastingly endanger the success of a system.

Limited perspective of the on-site customer. The on-site customer may not be a good representative for the needs of success-critical project stakeholders. Direct involvement in the negotiation process would ensure that the different viewpoints are appropriately captured.

Separation of concerns in decision-making. XP assumes that customers are solely responsible for business-related decisions while developers are responsible for technology-related issues. This separation in decision-making might lead to sub-optimal results as well as to a lower acceptance of the results.

In the next Section we will discuss negotiation techniques defined in EasyWinWin that address these limitations.

EASYWINWIN
The EasyWinWin requirements negotiation approach helps stakeholders to jointly discover, elaborate and negotiate their system and software requirements. EasyWinWin builds on the WinWin negotiation model and the WinWin spiral model developed by Barry Boehm et al. and extends the WinWin approach with a detailed negotiation process and groupware support. This section gives only a summary. For more details

refer to [5, 7, 8, 11] or visit the EasyWinWin website [6].

Negotiation Model
The WinWin negotiation model defines the basic elements that guide a requirements negotiation: *Win conditions* describe stakeholders' goals, *issues* capture known conflicts or constraints, *options* describe alternatives to overcome and resolve issues, and *agreements* denote mutually satisfactory solutions.

Major deliverables of a negotiation are (1) negotiation topics organized in a domain taxonomy, (2) definitions of key project terms, (3) agreements providing the foundation for further actions and plans, (4) open issues addressing constraints, conflicts, and known problems, as well as (5) further decision rationale showing the negotiation history (comments, win conditions, issues, options, etc.).

Negotiation Process
EasyWinWin defines a set of activities that guide stakeholders through a negotiation process. These negotiation activities are supported by group decision support tools (e.g., electronic brainstorming, polling, shared outliner, etc.). A detailed process guide is available that can be used by facilitators to moderate a negotiation [7].

Although EasyWinWin is extremely useful to develop a shared vision in the upfront phases of a project, it is also an integral part of the WinWin spiral model. Therefore, the activities described below are typically performed also in later iterations of a project.

The EasyWinWin process can also be applied without tool support, e.g., by using cards and boards in face-to-face meetings or email and telephone in distributed meetings. Our experience to date shows however that using collaborative tools is very helpful to deal with the complexity caused by the quantity of negotiation artifacts.

The activities in EasyWinWin are as follows:

Review and expand negotiation topics. Stakeholders tend to enter software development projects with vague, limited and retrospective perspectives. In this step they refine and customize an outline of negotiation topics serving as both an organizing framework and a negotiation checklist. Common high-level topics include features and services, interfaces to the end-user as well as software and hardware systems, system properties and level of service aspects, constraints like cost, schedule, development tools, and, support, and as well as system evolution concerns.

Brainstorm stakeholder interests. Stakeholders are encouraged to thoroughly explore their objectives using a brainstorming technique. If using a tool they make their contributions simultaneously and anonymously. They share different opinions, perspectives, and exploit the creative potential and experiences of the team. The brainstorming activity is thus an important technique to create a shared vision and to find possible metaphors.

Stakeholders identify goals for all identified negotiation topics and thereby explore less understood areas of the project, too. There are typically several brainstorming activities for the major negotiation topics identified upfront: Capabilities, interfaces, level of service, project & process, and evolution.

Converge on win conditions. Comments gathered in the brainstorming step are usually unstructured, redundant, ambiguous, and often vague. In this activity stakeholders jointly craft a list of clearly stated, unambiguous win conditions by considering all contributed ideas. Stakeholders also sort these win conditions into categories representing the major negotiation topics. An important challenge is not to miss any important ideas and at the same time weed out all pointless contributions.

Capture a glossary of terms. As participants build their win conditions, they use words that have special meanings within the context of a project or a domain. During the convergence step, the facilitator adds important terms to a shared list. Stakeholders use this information to create and jointly review definitions for these terms resulting in a glossary containing all key project terms. There are usually several rounds of reading and refining the definitions, but the time spent is worthwhile to reduce major risks stemming from miscommunication.

Prioritize Win Conditions. Polling in EasyWinWin is used (a) to determine priorities of win conditions, and (b) to reveal conflicts and different perceptions among stakeholders. Stakeholders rate each win condition for each of two criteria: Business importance shows the relevance of a win condition to project success; ease of realization indicates perceived technical or economic constraints of implementing a win condition. The basic rule during polling is "Vote what you know, don't vote what you don't know". So developers typically often focus on technical issues, while clients and users concentrate on the business relevance but this is not a strict rule. Win conditions are organized in one of four categories: "Low Hanging Fruits", "Important With Hurdles", "Maybe Later", and "Forget Them". The polling results are used during a negotiation by focusing on elements with high importance but they can also be used for other tasks, e.g., to define the increments in a project plan (see defining priorities in the planning game).

Reveal Issues and Constraints. Stakeholders examine the results of the prioritization poll to analyze patterns of agreement and disagreement. When using a polling tool win conditions with high/low consensus can be automatically identified. At the end of this step all win conditions with high instability in polling results are discussed until voting converges or issues resulting from different points of view are clearly understood and captured.

Identify Issues, Options, and Agreements. Conflict identification and resolution in the WinWin negotiation model is based on capturing *issues*, *options*, and *agreements*. Stakeholders identify these artifacts in several iterations and organize them in a hierarchy. The resulting deliverable is a hierarchy called the WinWin tree with win conditions, some of which will have issues as subheadings, some of which will have options as subheadings. The WinWin tree shows how agreements emerge in a negotiation process and thus captures the decision rationale. The group continues to work and tries to achieve a WinWin equilibrium, i.e., all win conditions and options are covered by agreements and there are no outstanding issues.

One main result of the negotiation process is a list of agreements and a list of open issues, which could not be resolved in the session. All unresolved issues (e.g., caused by stakeholder dissent) have to be managed as potential projects risks. Agreements of success-critical stakeholders are input to further planning activities and also elaborated into more precise requirements.

CONCLUSIONS AND FUTURE WORK

In this paper we discussed selected XP practices from the perspective of negotiation support and presented the EasyWinWin requirements negotiation methodology. Complementing XP practices with negotiation techniques such as defined by EasyWinWin has some important benefits:

Emphasis of shared vision. During an initial project meeting the success-critical stakeholders should meet and negotiate the strategic orientation of the new system. Using an approach like EasyWinWin would help to come up with a shared vision and mission statement for the project, and to constitute the general conditions under which the system has to be realized. This would facilitate the quest for the fundamental metaphor and the definition of high-level requirements supported by all involved stakeholders.

More complete stakeholder identification. The EasyWinWin approach supports identification and involvement of all success-critical stakeholders during the requirements negotiation process, which leads to (1) a higher acceptance of the new system (2) a lower rate of rework necessary caused by misunderstandings and misinterpretations of requirements.

Full perspective for on-site customer. During the release planning meeting the on-site customer is supported by all success-critical stakeholders from the customer side so they are able to refine the high-level requirements developed in the project planning meeting into more measurable statements. All stakeholders negotiate the requirements using EasyWinWin to agree on schedules and risks, as well as importance and feasibility of every feature – an essential input needed to assign the features to the next release.

Extensive stakeholder involvement in decision-making. Finally, EasyWinWin enables all stakeholders to negotiate and resolve business-related and technology-related issues. The focus on consensus leads to a higher acceptance of decisions and to an increased mutual understanding among the involved parties.

We are currently carrying out a case study to validate this approach under real-world characteristics. We will use the negotiation techniques in several XP projects. For example, we will explore the use of a lightweight EasyWinWin approach based on cards and boards in the XP planning game. This will also lead to an improved integration of XP guidelines and the EasyWinWin handbook.

REFERENCES

1. Beck, K. Embracing Change with Extreme Programming. *IEEE Computer,* Oct. 1999, pp. 70-77.
2. Beck, K. Extreme Programming Explained: Embrace Change. Addison-Wesley, 1999.
3. Beck, K. and Fowler, M. Planning Extreme Programming. Addison-Wesley, 2000.
4. Beck, K. One Team. On-line at http://groups. yahoo.com/group/extremeprogramming (Dec. 2001).
5. Boehm, B., Grünbacher, P. and Briggs, B. Developing Groupware for Requirements Negotiation: Lessons Learned. *IEEE Software*, May/June 2001, pp. 46-55.
6. EasyWinWin website: On-line at http://sunset.usc.edu /research/WINWIN/EasyWinWin (Dec. 2001).
7. Grünbacher, P. EasyWinWin OnLine: Moderator's Guidebook, A Methodology for Negotiating Software Requirements. USC-CSE 2000 & GroupSystems.com.
8. Grünbacher, P. and Briggs, B. Surfacing Tacit Knowledge in Requirements Negotiation: Experiences using EasyWinWin. *Proc. HICSS*, IEEE CS, 2001.
9. Highsmith, J. A. Adaptive Software Development - A Collaborative Approach to Managing Complex Systems. Dorset House, 2000.
10. Jeffries, R., Anderson, A., Hendrickson, C. Extreme Programming Installed. Addison-Wesley, 2000.
11. Stallinger, F. and Grünbacher, P. System Dynamics Modelling and Simulation of Collaborative Requirements Engineering. *Journal of Systems and Software*, Dec. 2001, pp. 311-321.
12. Wake, W. C. Extreme Programming Explored. Addison-Wesley, 2001.