Process Implications of Social Networking-Based Requirements Negotiation Tools

Nupul Kukreja
Center for Systems and Software Engineering (CSSE)
University of Southern California
Los Angeles, USA
nkukreja@usc.edu

Barry Boehm
Center for Systems and Software Engineering (CSSE)
University of Southern California
Los Angeles, USA
boehm@usc.edu

***Abstract*—Avoiding a major source of system and software project failures by finding more non-technical-user friendly methods of system definition and evolution has been a significant challenge. Five generations of the WinWin negotiation framework have improved such capabilities, but even the latest WikiWinWin toolset has encountered problems with non-technical stakeholder usage. With the advent of social networking and popularity of Facebook and Gmail, we have developed a radically different way for collaborative requirements management and negotiations. The new avatar of the WinWin framework called 'Winbook' is based on the social networking paradigm, similar to Facebook and content organization using color coded labels, similar to Gmail. Initial usage results on 14 small projects involving non-technical stakeholders have shown profound implications on the way requirements are negotiated and used through the system and software definition and development processes. Winbook has also been adopted as part of a project to bridge requirements and architecting for a major US government organization.**

***Keyword*—system definition processes; software evolution processes; collaborative requirements elicitation; requirements management; WinWin negotiations; social networking**

I. INTRODUCTION

A major source of failed systems and software projects has been the underrepresentation of users and other nontechnical stakeholders in the system definition and evolution process, as determined in the Standish Group surveys [12]. Traditional requirements engineering methods contributed to this problem by using uniformly-precise formalized requirements representations foreign to non-technical users. It was suggested in [1] to treat requirements as negotiated *win conditions,* in forms that were understandable by users, customers, developers, and other stakeholders, and to formalize these only where necessary.

Easy-to-use groupware for diverse stakeholder negotiation has been a continuing challenge [8, 9, 10]. Five generations of the WinWin negotiation framework have improved the capabilities [2, 7] but even the latest WikiWinWin toolset has encountered problems with non-technical stakeholder usage. With the advent of social networking and popularity of Facebook and Gmail, we have developed an innovative way for collaborative requirements management and negotiations. The new incarnation of the WinWin framework called 'Winbook' is based on the social networking paradigm, marrying the way people collaborate on Facebook and organize their emails using Gmail.

Winbook was developed after studying how people interact on Facebook and organize their emails using color coded labels in Gmail. There seemed to be a *fun* element when interacting with 'friends' on Facebook using *comments* and *likes* (similar to showing a thumbs-up in consent to the content 'posted' by someone). The user friendliness of being able to *tag* emails using color coded labels using Gmail seemed to make a strict folder-based organization of the emails obsolete, along with the capability to tag an email with more than one label. This seemed to echo well with the users. We decided to marry the concepts of social networking with email organization using labels to create a more interactive and an informal way to further the requirements engineering effort (and hopefully make it more 'fun' in the process).

The result was 'Winbook' – a tool written from ground up with the aforementioned capabilities along with the familiar look-n-feel of Facebook to enhance usability and facilitate quicker adoptability. Although Winbook uses a freeform conversational style for requirements elicitation, it has some enforced rigor via the WinWin equilibrium model for requirements negotiation [11].

II. A WINWIN NEGOTIATIONS PRIMER

In order to understand the process implications of social networking based requirements engineering, it is necessary to understand the WinWin methodology since it is the guiding thread for ideas outlined in this paper. The following steps form the crux of the WinWin methodology as discussed in [6]. The steps are not sequential, but interleaved as the stakeholders converge on a win-win equilibrium:

1) Refine and expand negotiation topics: Requirements deal with various stakeholder concerns (e.g. development, interfaces, properties, evolution etc). Stakeholders collaboratively elaborate a shared taxonomy of negotiation topics to help understand the scope of the project and the various aspects/concerns that need to be discussed.

2) Collect stakeholder win conditions: Most stakeholders have a rough idea of what they want. This step records the first draft statement of wants of the stakeholders. As a result the stakeholders learn what others want or expect from the system, enabling them to modify and clarify what they want from the system, by reading what others want.

3) Converge on win conditions: This step involves refining the win conditions to remove ambiguity and duplicates. A "shaper" role defined below expedites this.

4) Define a glossary of key terms: This step captures the domain vocabulary to develop a mutual understanding of language and eliminate ambiguity with respect to key concepts or terms.

5) Prioritize win conditions: Stakeholders rate the win conditions along either or both of the following criteria (based on their role in the project) –

a) Business Value – the degree to which the success of the project depends on the win condition

b) Ease of Realization – the degree to which a win condition is technologically, socially, politically or economically feasible.

6) Reveal issues and constraints: The above step of prioritization is used to provoke a constructive discussion on the possible issues/constraints that may be of concern to other participants. The 'variance' in the voting scale helps one see how their expectations stack up against the others and helps identify any unreasonable expectations, thus leading to a discussion of the areas of highest disagreement. This step helps make explicit any project constraints, assumptions or unshared information.

7) Identify issues and options: This step records any issues, uncertainties or risks that can be identified for each of the win conditions. For each of the issues, options for resolving that issue are also recorded.

8) Negotiate Agreements: Win conditions without any recorded issues are declared to be agreements. If issues exist, either the suggested options become the basis for agreement or further discussion is ensued – may lead to stakeholder revisions of win conditions or assumptions and constraints. When the group reaches an agreement acceptable to all, it is captured as an agreed-on win condition. When every win condition and every issue is closed by an agreement, 'WinWin equilibrium' is reached.

These steps define the WinWin negotiation framework and may be realized without any software tool per se. However, the benefits of a software tool are immense – they help capture the institutional memory of the negotiation effort as well as the mutually reached agreements along with the corresponding priorities (as per the ballots casted in step 5 above). This 'recorded' negotiation can then be used as a basis to understand the value propositions of the various stakeholders and provide impetus for a value focused prioritization, implementation and delivery of the software system.

III. WIKIWINWIN AND PROCESS IMPLICATIONS

As mentioned above, WikiWinWin was the most recent implementation of the WinWin negotiation framework. In this section we discuss the process implications of using WikiWinWin and its drawbacks that led to the development of Winbook.

WikiWinWin was based on the *twiki* platform [1] and implemented all steps of the methodology outlined above. The system was a linked network of pages – there was a page detailing each win condition, issue and option and separate pages for capturing the glossary, prioritization details and negotiated agreements. The agreements were recorded by physically updating each of the win condition (and/or option) pages, after conducting an oral negotiation. Creation of each of these pages required understanding the wiki syntax and the structure of the system for navigability. Separate training sessions were held for clients and students to help them come up the learning curve for using and documenting the results of the negotiations using WikiWinWin.

On average each team[2] would have 30-50 pages (for each of the win conditions, issues and options) in their project *twiki* – a huge maintenance and synchronization overhead. The twiki was customized to enforce the WinWin negotiation model for requirements negotiation, which made it necessary to have a page for each item – to facilitate easy search and navigation. The maintenance overhead outweighed the benefits in the long run.

A project 'homepage' existed that showed the entire set of win conditions, issues and options along with a color coded bullet indicating its agreement status – red (not agreed) and green (agreed). The 'homepage' was probably the most valuable view for the project team since they could see the equilibrium status at a glance.

The project teams along with their clients would initially brainstorm the desired set of capabilities, quality attributes, properties, constraints etc. that were expected from the software system (i.e. negotiation topics as in step 1 above). The negotiation sessions would proceed as outlined above and the items would be recorded using WikiWinWin. The project teams were encouraged to perform more iterations of prototyping after the initial negotiation sessions since they would have a better idea of what is or isn't valuable to their clients – allowing them to focus on the most valuable and/or risky items first.

The output of the WinWin negotiations along with the prototyping effort would lead to the creation of a system and software requirements document (SSRD). The SSRD template followed in the class consisted of identifying pre- and post-conditions, the requirement 'category' (i.e. user interface, levels of service etc.), priority, required inputs and sources, expected outputs and destinations etc. Each requirement in the SSRD was linked back to the corresponding win condition as recorded using WikiWinWin.

If a win condition was added or deleted, corresponding changes needed to be made to the SSRD. If a new requirement was identified and if it couldn't be mapped to any win condition, a new win condition was created in WikiWinWin and the SSRD would be updated accordingly.

[1] *Twiki.org*

[2] 7-8 member teams including students (6-7) and clients for the two-semester software engineering project course (CS577ab) taught at USC.

Going from requirements to architecture required creation and documentation of use cases in a system and software architecture document (SSAD). The use cases were similarly linked to the corresponding requirements. The added traceability led to a similar updating/syncing overhead as with win conditions and requirements.

As is evident, there was considerable manual effort involved in keeping the win conditions, requirements and use cases in sync to preserve *upstream traceability* and help percolate stakeholder value propositions all the way through implementation and delivery. Although the WikiWinWin tool with its wiki discussion threads was an improvement over previous WinWin tools [7], the student critiques often included dissatisfaction about the time & effort overhead when keeping the artifacts in sync and the difficulty of creating and synchronizing multiple pages in WikiWinWin. The clients, after the first negotiation session, would rarely use the tool for recording their win conditions – they would usually email updates to the project teams who would then enter them into the tool. Updating the win conditions, issues and options followed the same process. In general, after the initial win conditions were translated into the SSRD, the WikiWinWin tool was rarely used since changes were processed via the SSRD.

IV. WINBOOK CAPABILITIES

The WikiWinWin artifact synchronization difficulties caused us to look at new ways by which we could enhance the WinWin negotiation experience, reduce time & effort overhead in managing artifacts, capture institutional memory of the negotiation, and keep the principles of the WinWin methodology intact, all the while making it more of a *fun* activity. After studying the way people collaborate/interact on Facebook and organized their mails using Gmail, it was decided to experiment with a social networking based paradigm for the initial requirements engineering effort.

Existing social networking platforms like Facebook and Yammer are very generic in nature – primarily geared around a freeform conversational style of collaboration. Customizing them to enforce a WinWin negotiation framework for requirements elicitation proved to be a daunting task. Thus, it was decided to develop Winbook from scratch.

As mentioned, Winbook is a realization of the WinWin negotiation framework. However, the current version does not support capturing the shared domain vocabulary (as per step 4 above). Its look and feel is similar to that of Facebook – stakeholders *post* a win condition similar to how one posts status updates on Facebook. For each such win condition the *members* of that project are able to *raise issues or concerns* and also *suggest options* for resolving the issues similar to how one comments on their *friends' posts* – giving rise to a WinWin Tree structure (Fig 1.). Winbook also has *comment*ing capability, i.e. *members* can comment on the win conditions, issues and options for discussion purposes. The concepts of *likes* from Facebook was mapped to that of *agree* so that stakeholders could seamlessly 'signal' their agreement for the win conditions and/or options. Similarly the concept of *unlike* (where a user may revoke their prior

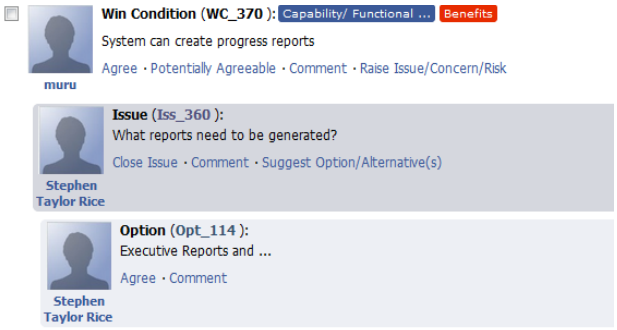

Figure 1. Zoomed in view showing the WinWin Tree.

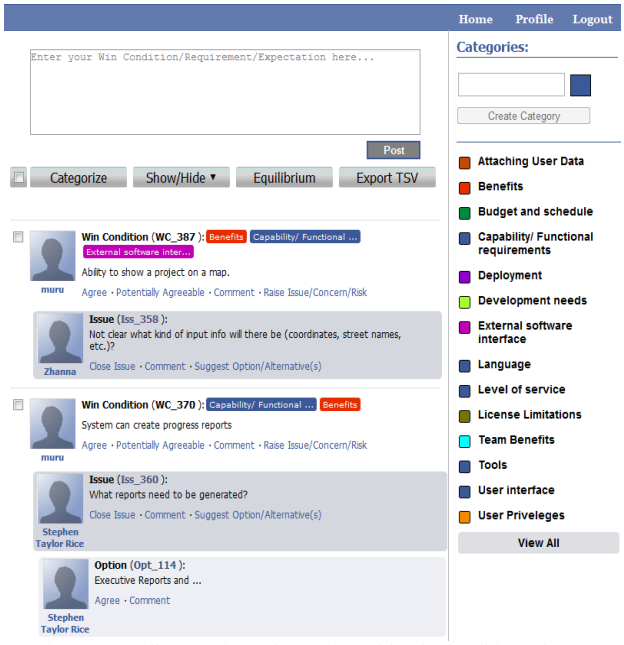

Figure 2. Wall Page view of a project with win conditions, issues, options and categories on the right

like) was mapped to that of *disagree* (i.e. a prior agreement may be revoked). The various posts i.e. win conditions, issues and options can be edited or deleted in place.

The win conditions are captured on a virtual *wall* (similar to a user's *wall* on Facebook) that the entire project team can access and update. The *posts* (win conditions, issues, options, comments) are displayed with the corresponding user's avatar (as selected during the sign-up process) to maintain the social networking look-n-feel. The *wall* serves as a virtual whiteboard for documenting user needs as win conditions (Fig 2.).

Winbook also has support for creating color coded *labels* similar to that of Gmail. Labels are called *categories* in Winbook and serve as the basis for creating negotiation topics (as per step 1 above). Win conditions can be tagged to belong to one or more categories. Color coding the category labels helps improve navigability.

Winbook has a special 'role' found valuable in previous WinWin version usage called the *shaper* – a team member who is responsible for 'shaping' the discussions and

shortcutting team-wide discussions of term definitions or overlapping artifacts. Once all team members cast their agreements for the win conditions and/or the corresponding options, the shaper casts his/her vote and the status of the win condition/option is upgraded to 'agreed'. The shaper may mark something as 'potentially agreed' if the team is researching the approach or is relatively confident of delivering on the expectation but needs more time or to signal the team that they may agree to the marked items. If a shaper doesn't cast his/her agreement the status of the win condition remains at 'not agreed'.

Winbook offers a color coded view of the notion of WinWin Equilibrium [11] – agreed win conditions or options are shown in green, those marked as potentially agreed as yellow and those not agreed are shown in red (Fig 3.). This color coded visualization of the *wall* helps see the agreement status at a glance and helps the clients know what the team is comfortable to committing and hopes to deliver on. Thus, the *wall* serves as color-coded negotiable contract where it's clear what are the expectations (win conditions), potential hurdles (issues), tactics for resolution (options), and what all can the team commit to delivering at their current level of understanding of the project.

Currently Winbook doesn't have an inbuilt prioritization capability[3]. However, win conditions are exported to a Google Spreadsheet where prioritization is performed along the two dimensions mentioned in step 5. The project teams (developers) estimate the 'ease of realization' of the win conditions using planning poker [13] and record their results in the spreadsheet. Having this data in a spreadsheet allowed them to run simple data analysis to see the overall priorities of the win conditions and draft out their planning activities along the most valuable items. The other benefit of exporting the win conditions as a spreadsheet was to have a seamless way of auto-generating an Excel-based SSRD[4] which may be used for furthering the software development effort – the SSRD was partially populated with the relevant win conditions and the traceability and prioritization information. The spreadsheet enabled easy cross-referencing and traceability within SSRD, alleviating the synchronization problems encountered with WikiWinWin.

V. Process Implications of Using Winbook

The teams continued to follow a similar process as detailed in Section III, i.e. WinWin negotiations coupled with risk/value focused prototyping led to the creation of an SSRD (partially auto-populated). The use cases documented in the SSAD were linked back to the corresponding requirements in the SSRD.

Although we did have an SSRD to start with, it soon became clear that content was just being maintained in two places with substantial overlap – Winbook and SSRD. This would still have some maintenance overhead even though some of the information was auto-populated. We observed

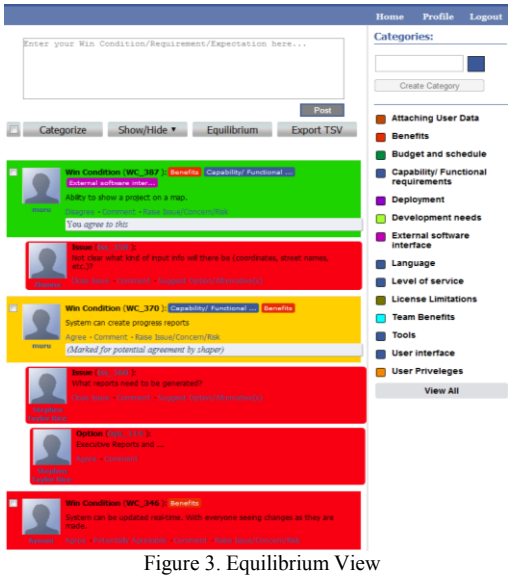

Figure 3. Equilibrium View

that the additional fields that need to be captured for each requirement (e.g. pre/post-conditions, priority etc.) were either self-evident or explained within the recorded Winbook negotiations. Thus, at least in the 14 small projects, we could have eliminated the SSRD altogether and have the use-cases in the SSAD directly link to the corresponding win conditions. In relation to agile methods, one can also elaborate the win conditions into user stories [14] and view the project *wall* as a product backlog with an added dimension of capturing potential risks/concerns and possible options for resolving them.

Another advantage of Winbook was having an up-to-date rationale capture of the stakeholders' needs along with sustained client requirements renegotiation, within the tool itself. You could in essence 'reprioritize as you go along' quite seamlessly. Winbook's capability of capturing issues or risks/concerns gave it an added dimension of considering various business and technical risks that may affect a particular win condition and making that explicit to all the stakeholders simultaneously.

We also observed improved effectiveness of milestone reviews as clients were more involved in the definition and prioritization of win conditions and everyone (project team and clients) was more rapidly on the same page. The project teams had a better understanding of the value propositions and would use that to channelize their prototyping efforts towards the riskiest and most valuable items first. The value focused negotiations and prioritization allowed the teams to plan for implementing and delivering the most valuable items first and managing their clients' expectations accordingly – since everything was easily accessible to everyone through the project *wall* and the *equilibrium* allowed everyone to see what the team was comfortable committing to (and the corresponding rationales for in case of not committing to a win condition or option).

[3] It was deferred owing to the start of the semester. It is currently being developed and will be integrated in the upcoming version.

[4] SSRD was previously an MS-Word document

VI. EVALUATION AND FEEDBACK

At the end of the semester each student submits their individual critique about the course – the processes and tools used, content, issues to consider and improvement suggestions. The common consensus was that Winbook was indeed helpful and lightweight for use and fairly easy to learn without extensive training. Playing planning poker to rate the ease of realization of win conditions helped manage expectations sooner by making them more explicit. The issues/concerns raised as part of the prioritization discussion helped identify underlying assumptions/constraints sooner. We even got a few responses stating that we should eliminate the SSRD altogether.

Most clients expressed their gratitude for having such an easy-to-use tool/framework which they could use with a relatively low learning curve and continually monitor the commitment status of the teams. There were many more clients directly using Winbook to post their win conditions or update existing ones than when using WikiWinWin. Some clients (e.g. the Los Angeles Department of Transportation and a South-Central Los Angeles community service organization) have expressed interest in having Winbook deployed at their site for internal use for brainstorming and organizing ideas – providing evidence of its ease of use and adoptability in practice. Winbook has also been adopted as a part of a project to enable a major US government organization to better bridge the gap between requirements and architecture, by using category labels for defining architectural elements and using the issues/options reporting capability to identify options for satisfying stakeholder win conditions.

We have also received a good deal of feature requests, most of which were similar to those of Facebook. The most common feature requests were to have prioritization 'integrated' into Winbook itself, have *status updates* similar to Facebook that would show a list of updates since a user's last login and most importantly the capability of having email notifications sent out by Winbook whenever the project *wall* was updated. These are the features that we are currently working on, along with several others for the upcoming student-project version of Winbook that would be deployed for the next class starting in Fall 2012.

VII. FUTURE DIRECTIONS

Based on the success and feedback of our trial use of Winbook with a social networking based way of capturing, negotiating and prioritizing the win conditions and tagging them with color coded category labels, we plan to extend it for doing end-to-end value based requirements gathering. The underlying architecture is flexible to allow one to create multiple *walls* per project. We plan to have another *wall* for explicitly capturing the expected benefits or goals/objectives of the project and have the capability of linking the win conditions directly to the end benefits thus facilitating goal oriented requirements elicitation and management.

We are currently prototyping the various capabilities and are using Winbook itself to help keep track of our *win conditions* along with raising issues/options for some of the complicated functionalities. We are effectively using Winbook to help enhance Winbook further. Various stakeholders have also requested features while they are using Winbook itself. The equilibrium capability helps everyone know which features *(win conditions)* are the ones that we would be committing to deliver next, along with a rationale for those we will be deferring. Winbook is thus showing itself useful for its own enhancement.

We will also explore how best to restructure the project processes and artifacts to have the negotiated win conditions serve as project requirements. This will include revised traceability relationships, and approaches for elaborating win conditions into stories or formal capability specifications (preconditions, post-conditions, inputs, outputs, sources, destinations, etc.); quality attribute elaborations (measurable, achievable, relevant, specific); and counterparts for interface, project, and evolution requirements.

REFERENCES

[1] B. W. Boehm, P. Bose, E. Horowitz and M. J. Lee, "Software Requirements as Negotiated Win Conditions," Proceedings of the First International Conference on Requirements Engineering, Colorado Springs, CO, April 18-22, 1994, pp. 74-83

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

[3] P. Grünbacher, S. Köszegi, S. Biffl, "Stakeholder Value Proposition Elicitation and Reconciliation", in Value-based Software Engineering (S. Biffl, A. Aurum, B.W Boehm, H. Erdogmus and P. Grünbacher, eds.), Springer Verlag, 2005, pp. 133-154.

[4] B. W. Boehm and H. Kitapci, "The WinWin Approach: Using a Requirements Negotiation Tool for Rationale Capture and Use", in A. Dutiot et al., eds., Rationale Management in Software Engineering, Springer 2006, pages 173–190.

[5] B.W. Boehm et al., "Using the WinWin Spiral Model: A Case Study," Computer, vol. 31, no. 7, 1998, pp. 33−44.

[6] D. Yang, D. Wu, S. Koolmanojwong, A. W. Brown and B. W. Boehm, "WikiWinWin: A Wiki Based System for Collaborative Requirements Negotiation," Proc. of the 41st Annual Hawaii International Conference on System Sciences, January 2008.

[7] D. Wu, D. Yang, B. Boehm, "Finding Success in Rapid Collaborative Requirements Negotiation Using Wiki and Shaper," Proceedings of the 43rd Annual Hawaii International Conference on System Sciences, January 2010.

[8] J. Herbsleb, "Global Software Engineering: The Future of Socio-Technical Coordination," in Future of Software Engineering (L. Briand and A. Wolf, eds.), IEEE PR2829, May 2007, pp. 188-198.

[9] J. Whitehead, "Collaboration in Software Engineering: A Roadmap," in Future of Software Engineering (L. Briand and A. Wolf, eds.), IEEE PR2829, May 2007, pp. 214-225.

[10] B. Cheng and J. Atlee, "Research Directions in Requirement Engineering," in Future of Software Engineering (L. Briand and A. Wolf, eds.), IEEE PR2829, May 2007, pp. 285-303

[11] B. W. Boehm and A. Jain, "An Initial Theory of Value-Based Software Engineering," in Value-based Software Engineering (S. Biffl, A. Aurum, B.W Boehm, H. Erdogmus and P. Grünbacher, eds.), Springer Verlag, 2005, pp. 15-37.

[12] J. Johnson, "My Life Is Failure", The Standish Group, 2006.

[13] J. Grenning, "Planning Poker," 2002, http://renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf

[14] K. Beck, "Extreme Programming Explained", Addison Wesley, 1999.