Comparing Software System Requirements Negotiation Patterns

Alexander Egyed and Barry Boehm
Center for Software Engineering
University of Southern California
Henry Salvatori Computer Science Building
Los Angeles, CA 90089-0781
{aegyed, boehm}@sunset.usc.edu

Abstract.
In a period of two years, two rather independent experiments were conducted at the University of Southern California (USC). In 1995, 23 three-person teams negotiated the requirements for a hypothetical library system. Then, in 1996, 14 six-person teams negotiated the requirements for real-world digital library systems.

A number of hypotheses were created to test how more realistic software projects differ from hypothetical ones. Other hypotheses address differences in uniformity and repeatability of negotiation processes and results. The results indicate that repeatability in 1996 was even harder to achieve then in 1995. Nevertheless, this paper presents some surprising commonalties between both years that indicate some areas of uniformity.

As such we found that the more realistic projects required more time to resolve conflicts and to identify options (alternatives) than the hypothetical ones. Further, the 1996 projects created more artifacts although they exhibited less artifact interconnectivity, implying a more divide and conquer negotiation approach. In terms of commonalties, we found that people factors such as experience did have effects onto negotiation patterns (especially in 1996), that users and customers were most significant (in terms of artifact creation) during the goal identification whereas the developers were more significant in identifying issues (conflicts) and options. We also found that both years exhibited some strange although similar disproportional stakeholder participation.

INTRODUCTION

In [Egyed-Boehm, 1997] we presented results of an analysis that addressed the question of whether people-intensive activities like software requirements negotiation are repeatable. This question was presented in the context of the SEI Capability Maturity Models (CMMs). The Software CMM identifies *Repeatability* as the goal for Level 2 processes [Paulk et al., 1995] and the Systems Engineering CMM's Level 2 goal is planned and tracked processes [Bate et al., 1995]. An early description of the Softwere CMM model [Humphrey, 1989] states:

> "Dr. W.E. Deming, in his work with the Japanese after World War II, applied the concepts of statistical process control to many of their industries. While there are important differences, these concepts are just as applicable to software as they are to producing consumer goods like cameras, television sets, or automobiles."

For some aspects of software development, such as problem report tracking, repeatability at the micro inch-pebble level may be as relevant as it is in producing consumer goods. For software requirements negotiation, macro-repeatability at the major milestone level is generally achievable and desirable. However, we concluded from our 1997 experience that micro inch-pebble repeatability for software requirements negotiation, if it can be achieved at all, would be very hard and would require strong proceduralization, which may lead to over-bureaucratization and which in turn may reduce the creativity of software designers. This result was based on an analysis of projects, in which we observed 23 three-person-teams while they negotiated the requirements for a hypothetical project. However, the main disadvantage of that analysis was that projects conducted by students as part of their coursework are only moderately representative of actual software development practice. Nevertheless, then we argued that such a setting could still serve as a lower bound on the repeatability scale for real software engineering processes and results.

To address the limitations of the hypothetical project environment, we conducted another analysis in 1996. There, we repeated a similar experiment in a real-client project environment were we observed 14 six-person-teams negotiating requirements of digital library applications for real customers and users of the USC Library. The following items summarize the key differences of the 1996 projects compared to the 1995 ones:

- Real customers and users
- Real vs. artificial conflicts to resolve
- Less negotiation pre-structuring, except for the use of a completeness checklist
- Complete development life-cycle performed (from inception to transition) [Boehm, 1996]
- Double team sizes, however, same basic customer, developer and user negotiation stakeholder roles

During the 1996 experiment we were also able to capture more detailed information about the negotiation process and its results. However, due to limitation in space we cannot present all of that here. If interested please refer to [Boehm-Egyed, 1998] which is entirely devoted to those projects.

In this paper we will focus on the analysis of commonalties and differences of both experiments and we will evaluate a number of hypotheses which have to do with the repeatability issue discussed above. These hypotheses and other observations are also put into context of the changed environment, which allows us to reason in what way hypothetical projects differ from real ones (and why).

THE PROJECTS

All projects in both years were in the digital library domain. Among other things, the tasks of the developer included the negotiation of the requirements of the proposed projects. Developers were either assigned to a project (all of 1995 and some in 1996) or they selected theirs themselves (mostly in 1996). During the negotiation process the team members assumed one of three stakeholder roles: developer, customer, or user. However, associated constraints with these roles differed between the years as described later. The following summarizes the projects.

1995 Projects

In 1995, 23 three-member student teams negotiated requirements for a hypothetical library system called SDI (Selective Dissemination of Information). The target environment was a hypothetical university with three campuses. The goal was to integrate the libraries of these campuses and provide additional services like user interest profiling (compare new acquisitions with user profile and notify user if match is found). Additional monetary and schedule constraints existed.

The basic components for the SDI system were defined in advance and estimations of their individual sizes in lines of code were provided. The students had the freedom to decide whether to incorporate those components and if yes in what level of detail. The latter allowed the students to choose different designs of the components which all provided basic capabilities and some additional ‘nice-to-have’ features.

Besides choosing software components, the students had to make a few hardware choices as well. For instance, they had to choose the type of server, with different speed-risk tradeoffs. The students were, however, not asked to do the actual coding of the system. They only had to perform the initial stages of the development process, from requirements negotiation to high-level design. The projects ended after roughly 8 weeks.

To compensate for the lack of real customers, students were given special negotiation goals depending on the role they were playing. As described above, there were three stakeholder roles (customer, developer, and user) and each member of the team had to take one of these roles. The special goals just mentioned were provided with the intent to create some ‘artificial conflicts’ which the stakeholders had to master in order to complete the negotiation part of the development cycle. For instance, the customer was burdened with a very tight cost constraint, the user got a list of his or her ‘must have’ items, and the developer was asked to minimize cost and schedule risks. For more detailed information about the project environment and the involved people refer to [Egyed-Boehm, 1997]

1996 Projects

In 1996, 14 six-member teams set out to do a very similar task of developing components for an integrated library system. Instead of being given a project, these teams selected their topic out of a wide range of (mostly multimedia related) projects (see Table 1). Each project was conceived by a real customer from the USC Library and was derived out of a need in his/her community. Besides proposing the system, the library customers were also involved in negotiating the requirements with their student team(s) (a few customers initiated two topics or administered two teams working on the same topic). Further, the projects were planned, designed and implemented over a period of

Table 1: Project topics

1995	1996
Library SDI System Selective Dissemination of Information	Stereoscopic Slides
	Latin American Pamphlets
	EDGAR Corporate Data
	Hancock Image Archive
	Interactive TV Material
	Technical Reports
	Cinema-TV Moving Images
	Maps
	Searchable Archives for Images
	Korean-American Museum
	Planning Documents
	Medieval Manuscripts

two semesters, with the ultimate goal of delivering a real product with sufficient initial capabilities at the end of the second semester.

Even though most projects were different from each other, many were built within a similar domain. Most products had to provide some form of user and administrator interfaces on top of a multimedia database. Further, all projects were required to be developed using the World Wide Web as an interface (browser).

Since neither customer nor developer had detailed ideas on what the system should really look like, the 1996 projects were much less restrictive than the 1995 ones. In fact, there were a few cases where student teams actually came up with solutions which did not only solve the (intermediate customer) problem but also improved the entire business process much to the delight of the customers. Nevertheless, this new form of freedom brought also problems with it, like fuzzy requirements, personnel and resource conflicts, etc.

Like the 1995 projects, the students had to find more than just software solutions [Boehm, 1994]. They had to integrate new multimedia concepts into existing library hardware and COTS packages (e.g. SIRSI – the library information system). Further, they had to find procedures to deal with 'non computer science problems' like copyright issues or even how to scan fragile manuscripts.

Furthermore, these teams were also subdivided into the three basic stakeholder roles for the purpose of negotiating requirements (two students per role). The students who played the customer and user roles served as mediators to the real library customers and users since these people were not always available or computer literate enough to use our tool.

In 1996, no additional artificial constraints were defined for the stakeholder roles. The real constraints were difficult enough. However, since the projects were required to be carried out in much more detail, we defined rules on who should become a developer, customer or user based on the development roles they were playing. So for instance, the architect and prototype people represented the developer, the operational concept and requirements people represented users, and the team leader and life-cycle plan people represented the customers in the requirements negotiation.

At the end of the negotiation, above stakeholders assumed new roles as developers; each person with distinct primary responsibilities as described above (e.g. requirements, architecture, etc.). For more detailed information about the project environment and the involved people refer to [Boehm-Egyed, 1998] and [Boehm et al., 1998].

Although, this project environment constitutes a real development environment in many ways it may not apply in all situations (e.g. the issue of cost and payment was not very relevant in our projects – industrial projects however do not have that luxury; further, our team sizes were also comparatively small). Nevertheless, we belief that in most other aspects our development projects did not differ from industrial ones.

OUTLINE

In the following, this paper will cover the following items:

- Present the *WinWin Development Model* which all teams used to conduct their negotiation.
- Present the relevant results of both 1995 and 1996 experiments. Due to limitation in space we defer some of the details to the individual experiment papers of 1995 [Egyed-Boehm, 1997] and 1996 [Boehm-Egyed, 1998].

- Compare both experiments in order to point out their commonalties and differences with respect to the 'key characteristics' described above.

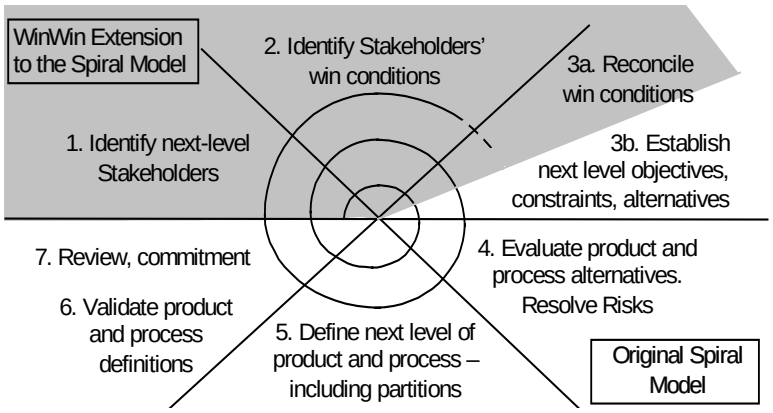
WINWIN DEVELOPMENT MODEL

The WinWin development model incorporates a number of basic models; the *WinWin Spiral Model [Boehm, 1988][Boehm,1996]*, the *WinWin Negotiation Model [Boehm et al., 1995][Lee, 1996]*, *COCOMO [Boehm, 1981]*, and others. It is beyond the scope of this paper to address all of them. In the following, we will therefore only describe the WinWin spiral model and WinWin negotiation model since many of the results presented in this paper are based on the concepts incorporated in those model.

The WinWin development model is based on Theory W [Boehm-Ross, 1989][Fisher-Ury, 1981], a systems management theory. Theory W provides the principle and criterion that your project will succeed if and only if you make winners of all the critical stakeholders. Clearly, if all the stakeholders (users, customers, developers, maintainers, interoperators, etc.) are delighted with the project outcome, your project has succeeded. However, if you create a win-lose situation, the losing stakeholder can generally work out a way to survive at the expense of the other stakeholders, producing an unsuccessful project. For example, if a customer and user drive a developer to underbid a set of ambitious requirements, the developer can generally reinterpret the requirements in a way that is contractually compliant, but which leaves the customer and user with a lower value system.

Figure 1 shows the integration of Theory W concepts into the Spiral Model, resulted in the *WinWin Spiral Model*. The spiral model approach [Boehm, 1988] incorporates an evolutionary process which allows, and even encourages, multiple iterations of system and software development stages until the final goal is achieved. The Theory W extension modifies the spiral model in such a way that each iteration identifies all critical stakeholders and their win conditions in advance. Thus, three activities were added upfront:

Figure 1: WinWin Spiral Model [Boehm,1996]

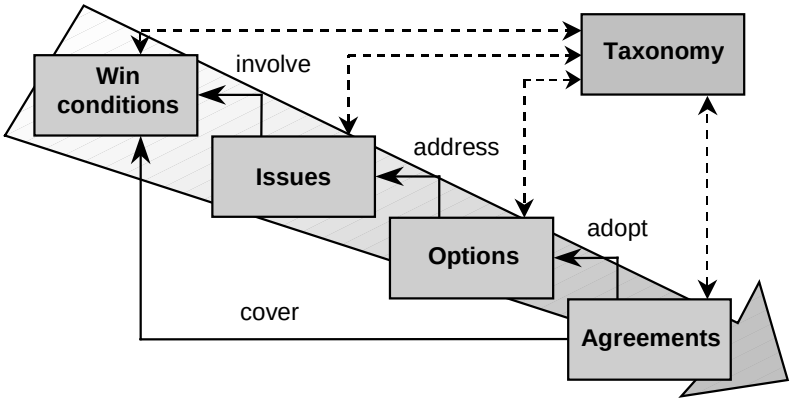


- Identify the system or subsystem's key stakeholders
- Identify the stakeholders' win conditions for the system or subsystem
- Negotiate win-win reconciliation of the stakeholders' win conditions.

Compared to hierarchical sequentialized systems engineering processes such as SPC's IDEF0-based Integrated Systems and Software Engineering Process [Rose et al., 1996], the WinWin spiral model emphasizes cycles of concurrent elaboration. In each cycle, the key system definition artifacts (operational concept description, system requirements, system prototypes, and life cycle plan) are elaborated in a risk-driven fashion, along with a system rationale artifact which establishes the feasibility and consistency of the other artifacts.

The WinWin spiral model also added the concept of "anchor points" to synchronize the spiral cycles with critical management decision points. The front-end anchor points are the Life Cycle Objectives (LCO) milestone and Life Cycle Architecture (LCA) milestone. [Boehm, 1996] provides more detail on how these milestones relate to spiral cycles in various situations. For system engineering and architecting of hardware-software systems [Rechtin-Maier, 1997] provide a similar synchronizing mechanism between software spiral cycles and intermediate hardware configurations, represented as intermediate circles in a spiral diagram.

Figure 2: The WinWin Negotiation Model



To complement the spiral process model, a negotiation support has been developed (the WinWin Negotiation Model) to ensure that the right amount of communication and collaboration is guaranteed during all cycles of the system development process. All teams in both years used this model and its supporting tool the *WinWin System* [Boehm et al., 1995][Horowitz, 1996] to negotiate the requirements for their respective system.

The WinWin Negotiation Model [Boehm et al., 1994] is based on four artifact types: *Win Conditions*, *Issues*, *Options* and *Agreements* (see Figure 2). Win conditions capture the stakeholder goals and concerns with respect to a new system. If a win condition is non-controversial, it is adopted by an agreement. Otherwise, an issue artifact is created to record the resulting conflict among win conditions. Options allow stakeholders to suggest alternative solutions, which address issues. Finally, agreements may be used to adopt an option, which in turn resolves the issue.

The WinWin model also includes a tailorable Domain Taxonomy, which enables Stakeholders to link artifacts to taxonomy items and to access those artifacts via the taxonomy. In case of the 1996 projects, the taxonomy structure followed closely the table of contents of the requirements documents which was tailored towards multimedia library applications (see Table 2). Thus, the negotiators used the taxonomy as a checklist for sufficient coverage. The 1995 teams used the taxonomy in a similar context. However, in their case the negotiation guidance was not primarily meant to achieve sufficient coverage of taxonomy items but it was meant to resolve the built-in conflicts in the hypothetical SDI application.

Table 2: Simplified Taxonomy

1 Media operations	4 Quality
1.1 Query/Search/Browse	4.1 Response Time
1.2 Access Control	4.2 Reliability
1.3 Audio/Video Operation	4.3 Security
1.4 Update/Input	4.4 Usability
1.9 Others	4.5 Interoperability
2 Interface	4.6 Workload
2.1 COTS (SIRSI, etc.)	4.7 Cost
2.2 Database (File Access)	4.8 Schedule
2.3 User/Admin. Interface	4.9 Others
2.9 Others	5 Operations/Environment
3 Administration	6 Development/Process
3.1 User Management	7 Evolution/Maintenance
3.2 Usage Monitoring	9 Others
3.9 Others	

The WinWin System is a telecooperation tool that was built to support the WinWin negotiation model. The tool uses Inter- and Intranet support to enable collaboration between distributed stakeholders. It may be used both synchronously and asynchronously, meaning that stakeholders may negotiate using the tool at the same time, but they may also use it at different times. Further, a number of support tools are integrated with WinWin to assist in the negotiation, especially in order to support tradeoff analyses, and to identify and resolve risks. The following are a few examples:

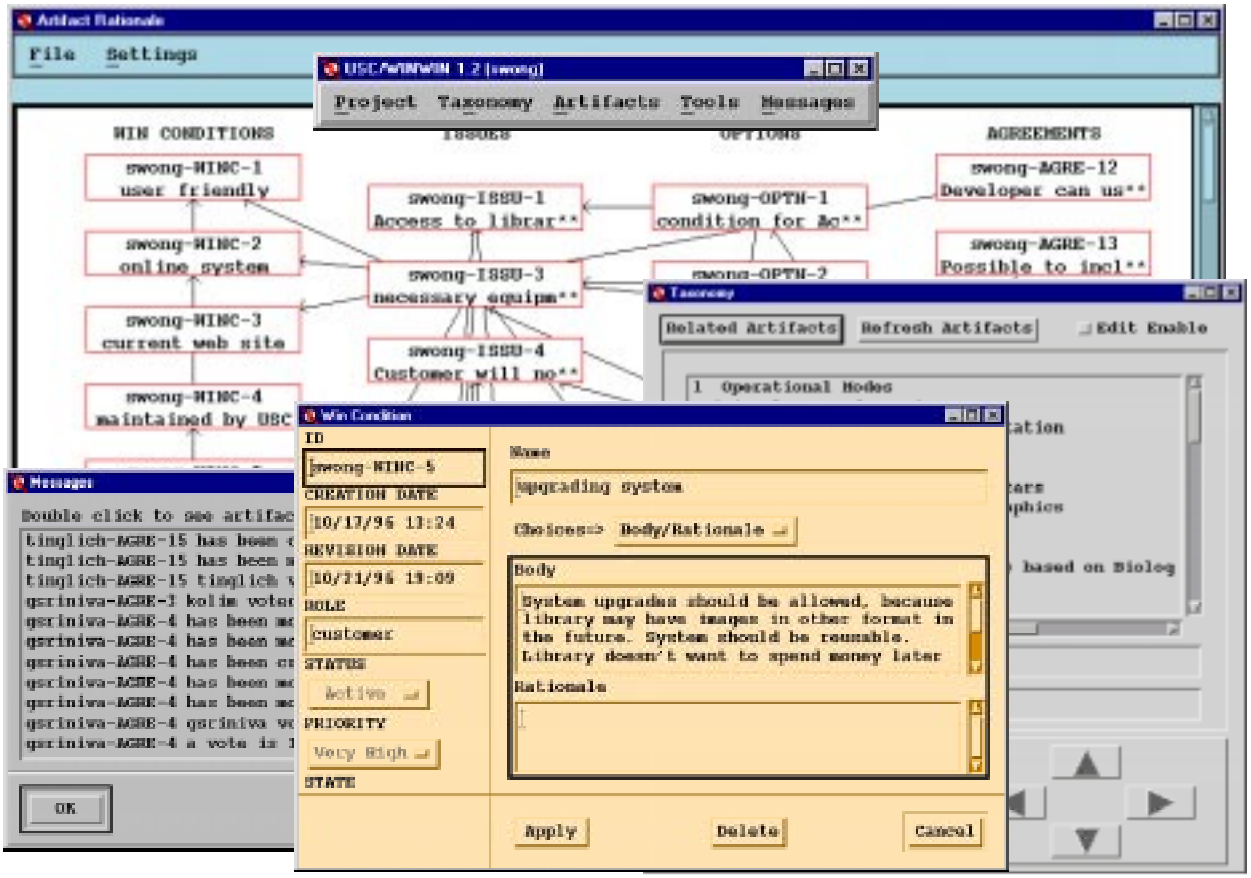


Figure 3: The WinWin Tool

Table 3: WinWin System Capabilities

Capability	WinWin System
Components	Win Conditions, Issues, Options, Agreements, Terms, Taxonomy Items
Connectors	Connections between Artifacts: • *Cover*, *Resolve*, *Adopt* (e.g. option resolves issue) • Simple *Relates To* showing some inter/intra artifact dependency • *Replace* (e.g. Agreement replaces an older one) Connections between Artifacts and Taxonomy Connections with external Tools (e.g. Analysis tools)
Views	Taxonomy view reflecting domain categorization Rationale view reflecting dependency and decision tree Message views reflecting the change history
Navigation	Hypertext Style browsing between Artifacts, Messages, and Taxonomy
Change History	Implicit through Artifact types Explicit through Messages describing nature and extend of changes
Information Sharing	Semi-automatic update (update only when requested by user)
Security	Artifact ownership; artifacts are frozen once voting is initiated
Completeness	Taxonomy (domain coverage) Artifact Flags (e.g. issues are unresolved/resolved; options are unused/used; etc.)
Group Control / Collaboration	Artifacts, Messages, and Comments Taxonomy, Terms, Rationale Graph, Status Summary External tools (Attachments)

- A4 (Architecture Attribute Analysis Aid): Architecture-based analysis of cost, schedule, performance, and reliability.
- Rapide: A architecture tool for modeling and simulating systems and identifying problems (deadlocks, bottlenecks, etc.) in the architecture.
- COCOMO (Constructive Cost Model) II: Cost/Schedule estimation tool [Boehm, 1981].

A screen hardcopy of the WinWin tool is given in Figure 3. In the foreground, a customer win condition of one of the Library projects is visible. To the right is the Taxonomy window and to the left the Message window. This window contains short descriptions of changes in the order they were made. The background of the figure contains a graphical box-and-arrow diagram reflecting the current state of artifacts and their connections. Table 3 summarizes the tool's capabilities.

LIMITATIONS

This paper compensates for some of the limitations of the 1995 study because some of those limitations had to do with the fact that the project environment was more hypothetical. Nevertheless, a few limitations remain: Although, the 1996 data collection was quite comprehensive, the 1995 study lacked some of the data. Thus, we can only compare data that was captured by both studies.

PROJECT ANALYSIS

As mentioned earlier, during both years, the same overall development process (spiral model) was followed, the same negotiation tool (WinWin System) was used, and the same people were doing the analysis of the findings. Thus, the comparison is less blurred by fundamental differences like terminology, process, etc. The following summarizes the types of hypotheses which were formulated about the individual experiments and the comparison. The experiment hypotheses are for the most part about uniformity, repeatability and negotiation results across the 23/14 teams. The comparison hypotheses are mostly generalizing those experiment hypotheses in order to allow meaningful comparisons. Hypotheses were investigated in the following categories:

- Negotiation Cardinality (syntactic patterns)
- Effects of People
- Negotiation Schedule and Process

Some of the hypotheses in these categories were assessed quantitatively based on the information captured by the WinWin tool through its instrumentation. Other hypotheses were investigated using more qualitative means of evaluation like questionnaires, reviews, etc.

Negotiation Cardinality

A series of mostly quantitative hypotheses about the negotiation results can be investigated by looking at the cardinality of negotiation elements like artifacts, comments, relations, etc. For instance, a uniform solution path would imply a uniform number of artifacts for each type and a similar way of connecting them.

Hypothesis A1: Number of Artifacts, Comments, and Connections will be similar (in number or ratio). Like in 1995 the hypothesis that *the number of artifacts, connections, comments, etc. created by the stakeholders would be similar for all teams (±10%)* was rejected in 1996 as well. The average, minimum, and maximum number of artifacts, comments, and connections were even more diverse in 1996.

To get more insight, artifact types were evaluated in more detail (see Figure 4). Here we find strong similarities. We were able to accept the hypothesis that *win conditions would be the most common artifact type because they represent the knowledge base and that there would be more options than issues* for both years. Agreements, however, turned out to be exceptional. In 1996 the agreement vs. win condition ratio was much higher than the corresponding one in 1995. This may be attributed to the fact that the 1996 projects were asked to use the Library Multimedia domain taxonomy as a checklist for formulating win conditions and agreements. The average number of win conditions per team was also considerably higher in 1996 (513/14=37 vs. 402/23=17).

Hypothesis A2: Most Win Conditions will be non-controversial. As indicated in Table 4 less than half of the win conditions (242 of 513) across all 1996 teams were involved in issues. In 1995 the number is higher with about 60%

Figure 4: Number of Artifacts per Type

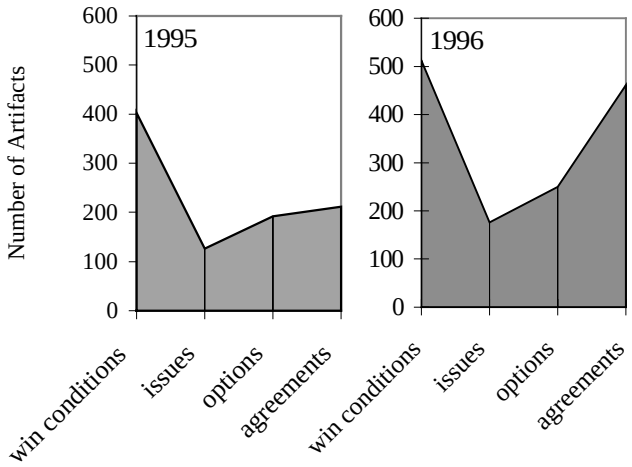


Table 4: Connectivity Complexity

Conditions	1995	1996
Win conditions covered by issues	246 out of 402	242 out of 513
One option per issue	55 out of 121	123 out of 179
More than one option per issue	66 out of 121	56 out of 179

of all win conditions being involved in issues (246 of 402). Thus, although the hypothesis is accepted for the real 1996 projects, it is rejected for the 1995 projects, which were structured to emphasize conflicts.

Hypothesis A3: Most issues will be straightforward to resolve. Like in the hypothesis above, 1995 projects were a bit more complex in that they had more complex options than the 1996 projects. Again we accept this hypothesis for the real 1996 projects but reject it for the artificial 1995 projects (see again Table 4).

Other hypotheses about this category, defined in [Egyed-Boehm, 1997] and [Boehm-Egyed, 1998], show further commonalties and discrepancies in cardinality of the same nature as presented above. Generally, the 1995 projects exhibited more complex artifact interrelationships. The higher complexity in 1995 may have been caused by a few strong constraints placed on the projects. For instance, they had to deal with a very tight cost constraint, which had side effects throughout the entire negotiation and on many artifacts. Overall, the 1995 projects were focused on conflict resolution. The 1996 projects were focused more on completeness of requirements, via the use of a domain taxonomy and a checklist for win conditions and agreements. For the more realistic projects (1996), the main conclusions were that most requirements were non-controversial and that most issues were simple to resolve.

Effects of People

Of high interest was also whether the stakeholders showed similar unusual patterns in the way they interacted and collaborated and how this was affected by people factors like experience. We further wished to verify whether the stakeholders would exhibit similar 'role behaviors' [Bullen-Bennett, 1990].

Hypothesis B1: Artifact Contribution will vary by Roles. In 1995 it was assumed that *all stakeholders within all teams would participate equally during all 'phases' of the negotiation […] regardless of artifact type.* Figure 5 shows the absolute number of artifacts for both 1995 (left side) and 1996 (right side). In 1995 it was found that users had almost only half the number of artifacts the customers had. More detailed investigation revealed that this was also true for most teams individually. It was very surprising to see that the 1996 experiment resulted in a very similar stakeholder ranking. Again, the user had much fewer artifacts than the customer; however, the ratio is not as extreme as in 1995. Thus we reject above hypothesis because uniformity of artifacts is not given for both years. Either the

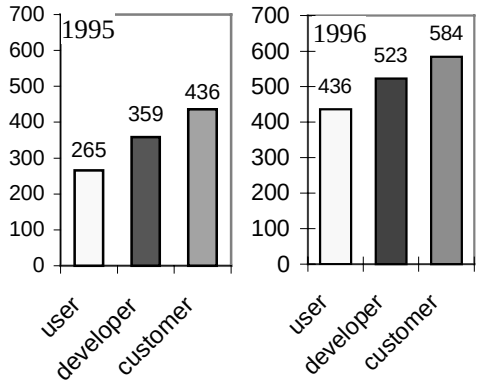
customer or the developer had more artifacts than the user with high significance (exceptions were 2 out of 23 in 1995 and 1 out of 14 in 1996; see also Table 5).

Table 5: Comparison of number of artifacts between types of stakeholders

1995: Number of Artifacts	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Customer > User	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓		✓
Developer > User	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓		✓		✓
Customer > Developer				✓	✓			✓	✓	✓			✓	✓	✓			✓					✓
Customer or Developer > User	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓		✓

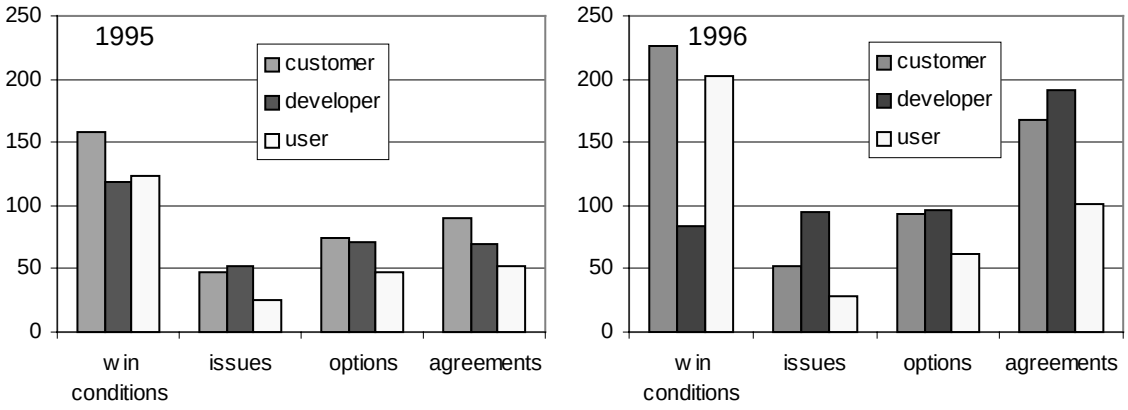
1996: Number of Artifacts	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Customer > User	✓		✓	✓			✓	✓	✓		✓	✓	✓	✓
Developer > User		✓		✓	✓		✓	✓	✓	✓	✓		✓	
Customer > Developer	✓		✓	✓		✓	✓	✓	✓			✓		✓
Customer or Developer > User	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓

Figure 5: Number of Artifacts



Hypothesis B2: Artifact Contribution will vary by Types. In 1996, it was found that users and customers originated more win conditions, developers originated the most issues, and developers and customers originated more options and agreements. This indicated that customers and users were more important during goal identification and developers were more important during risk (issue) identification and resolution. Comparing these figures with 1995 we identified a similar pattern. Figure 6 shows the number artifacts per role and type. The numbers on the left are from 1995 and the numbers on the right are from 1996. Note the similar patterns in both years.

Figure 6: Number of Artifacts per Roles and Artifact



Hypothesis B3: More experienced teams will have some different negotiation characteristics. In 1996 the more experienced teams usually completed their project negotiations in a shorter amount of time and also applied a more iterative negotiation approach by introducing artifacts long after voting had started (many other teams did not do that). We found that 1995 projects did not show these trends. This may be due to strong differences between both years in negotiation schedule and process, as it will be discussed in the next section.

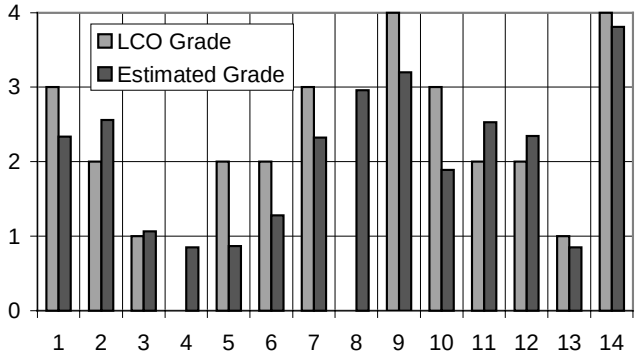
Hypothesis B4: *Some negotiation patterns will correlate with more effective negotiation results* (quality is measured by LCO package grades). It turned out that the two negotiation approach features *iterations* and *productivity* as well as the team experience were found to be a strong predictor for the LCO grade. The LCO package marked the end of the first life cycle iteration roughly two weeks after the negotiation had ended. Table 6 shows the grades of the LCO (only for 1996) packages as well as the predictor variables that were found to be significant (Team 8 is the only exceptional case in 1996). The resulting correlation between estimated grade and actual grade is 0.82, which can be considered highly significant (see Figure 7). Unfortunatelly we do not have comparative data for 1995. Nevertheless, we still wanted to present this result in this comparsion since the strongest quality driver here was team experience which once again points out, how important this factor is. We also list the 1995 characterisitcs here so that the interested reader may be able to compare them with other data presented in this paper.

Table 6: Some Team Characteristics

1995 Projects	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
LCO Grade	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Experience	M	VL	VL	VH	VL	VL	VL	VL	M	VL	H	L	L	H	VL	M	M	L	VL	VH	L	VL	L
Iterations	L	L	L	VH	L	VH	L	VH	L	L	VL	VL	L	L	L	L	L	L	M	VH	L	VL	L
Number of artifacts	VL	L	M	M	M	H	M	H	VH	VH	L	M	L	VH	VH	VL	L	VL	H	H	VL	VL	L

1996 Projects	1	2	3	4	5	6	7	8	9	10	11	12	13	14
LCO Grade	H	M	L	VL	M	M	H	VL	VH	H	M	M	L	VH
Experience	L	M	VL	VL	VL	M	L	M	H	L	M	M	VL	M
Iterations	M	L	L	L	VL	VL	H	H	L	H	H	L	L	VH
Number of artifacts	L	M	H	L	VH	VL	M	VL	M	H	VL	VL	VL	VH

Figure 7: Actual vs. Estimated LCO Grade



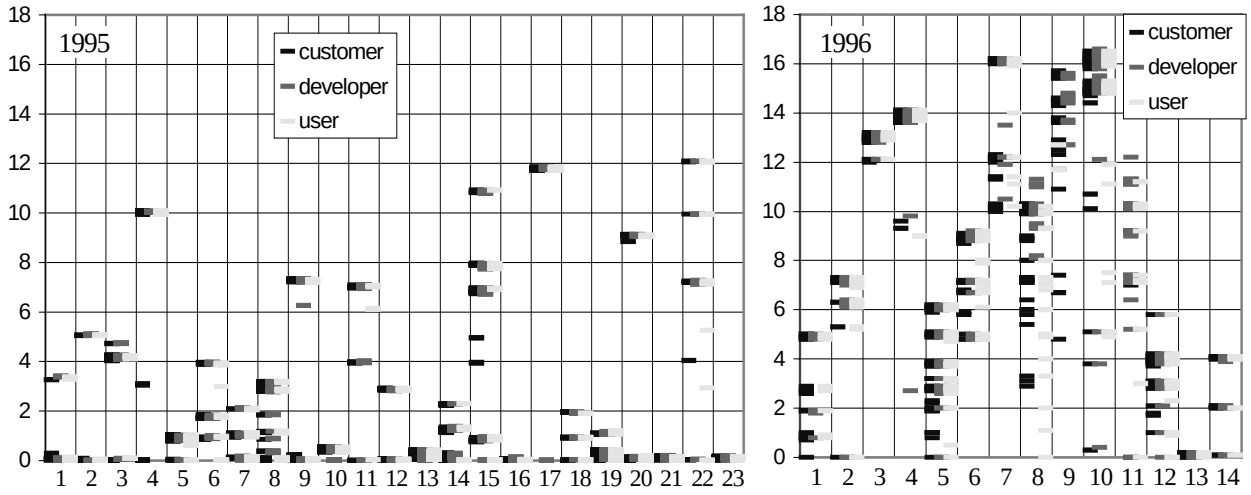
Unfortunately, we cannot elaborate on other people factors here since we only started capturing sufficient information for that in 1996. For instance, in 1996 we were able to capture people 'satisfaction' of both customer (user) and developer, which showed some interesting patterns [Boehm-Egyed, 1998]. For example, WinWin was found to promote cooperativeness, focus teams on key issues, but would have been more effective with better preparation and concurrent prototyping.

Negotiation Schedule and Process

This section will investigate the negotiation process and schedule of teams in both years. For instance, we wished to investigate whether both years would exhibit some similar negotiation process patterns.

Hypothesis C1: Average number of iterations to resolve artifacts will be similar. The number of iterations turned out to be much higher in 1996 than the ones in 1995. This issue was addressed in a previous hypothesis (C1) as well and

Figure 8: Artifact Creation and Revision Table

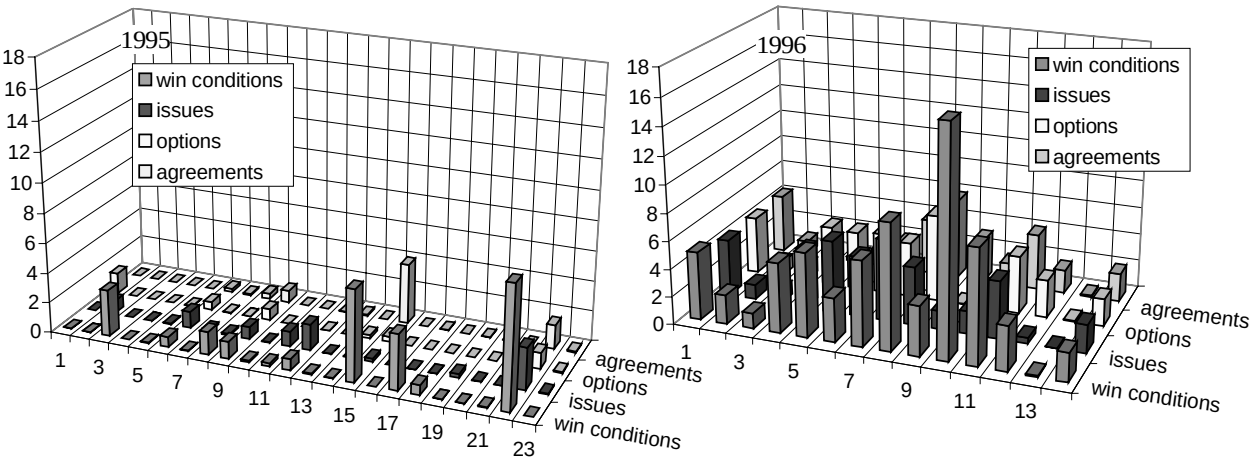


is further supported by Figure 8. Figure 8 shows the creation and revision of artifacts for each team over a period of 18 days when the stakeholders used the WinWin tool. For instance, all stakeholders of team 1 (year 1996) used the tool on day 1, 2, and 5. On day 3 it was used by the customer and user only and on day 4 it was not used at all. We assumed that *the negotiation problems were rather simple and would require only one or two sessions to solve them*. This was mostly true for 1995 but we found that this hypothesis does not apply for 1996. Clearly, most teams in 1996 were not able to come up with a solution immediately (in the first session) but had to conduct multiple iterations over a period of almost 18 days.

Hypothesis C2: Creation times of artifacts per role will be similar. Based on the role behavior patterns we described in section B, we wished to identify whether different stakeholders participated at different times during the negotiation. This hypothesis goes hand in hand with the hypothesis whether *the tool was used synchronously (at the same time) or asynchronously* (see Figure 8). We found that stakeholders used the tool synchronously for the most time. This observation is also related to the finding that most stakeholders were involved (created or revised artifacts) from the beginning until the end.

Hypothesis C3: Artifacts will be revised with similar frequency. The 1995 teams produced in average only half as many artifacts but as it can also be seen in Figure 9 they spent less time (schedule) per artifact. It was therefore assumed that *the average time from artifact creation to artifact revision would be similar between both years*. This hypothesis turned out to be very wrong and had to be rejected. Figure 9 shows that 1995 projects rarely ever modified the same artifact for more than 1 or 2 days in average for all types. For instance, none of team 23's artifacts

Figure 9: Average Duration to resolve Artifacts (first creation vs. last revision)



were revised after they were created. Team 22, on the other hand, still made modifications to win conditions in average 8 days after they were created. This implies that team 22 was much more thorough in revisiting and adapting artifacts (also implies a more iterative negotiation process).

In contrast to the 1995 projects, 1996 project artifacts were usually modified over a much longer time frame. This may be an indication that the higher degree of freedom in 1996 made it also very hard to find and resolve problems. The need for increased duration may also have been caused by the higher amount of artifacts produced (1996 project had more than twice as many artifacts). If this were the case, it would be reasonable to assume that 1996 projects also resulted in more complex artifact dependencies. Hypothesis C1 seemed to reflect this as well.

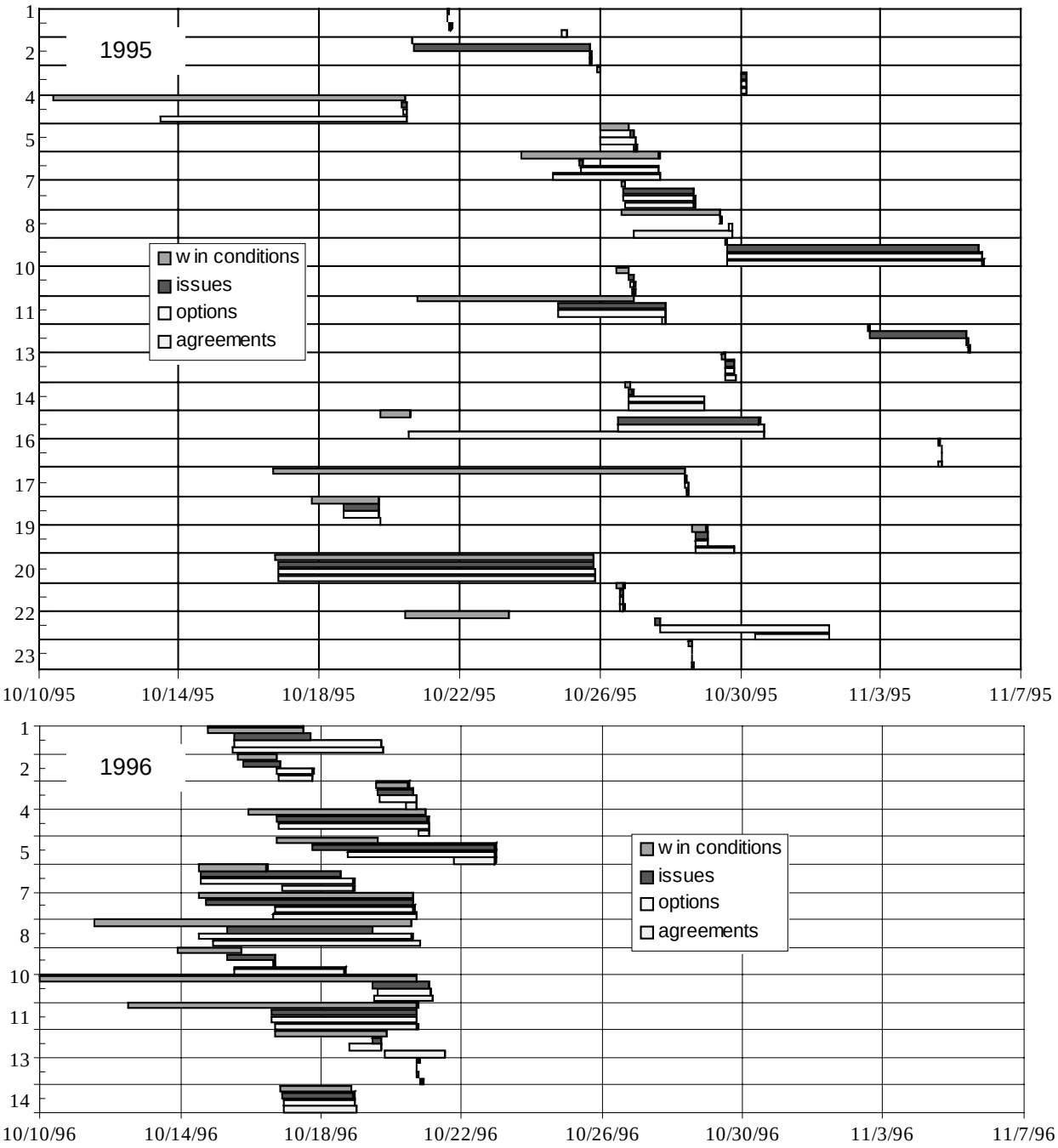


Figure 10: Creation time table for artifacts

Hypothesis C4: Creation time of artifacts of different types will be similar. It was expected that *agreements would be created as soon as a few win conditions were entered.* This was found to be wrong for a large number of teams in both years. Many teams started to create agreements only after most if not all win conditions were entered (implies a more waterfall like negotiation process). Further it was also found that both years the teams made rather poor use of time. Most teams used the WinWin tool only for a rather brief period in this multiple week activity, even less so in 1995 (see Figure 10).

Hypothesis C5: Average time to resolve artifacts will be similar (duration from problem to solution). In C2 we wished to evaluate the time duration an artifact got modified. On the other hand, here we wished to know how much time it would take to actually resolve an artifact. Thus, starting from its creation, how much time would be spent until it is *resolved* by a *passed* agreement (from problem to solution). Again, both years exhibited strong differences except for win conditions, which were more uniform (see Figure 11). Most obvious are the differences in issues and options. It took less than a third of the time to resolve them in 1995 compared to 1996. On the other hand, it took somewhat more time to resolve 1995 agreements than the 1996 ones. Thus, we have to reject similarities between both years except for win conditions. Altogether, the average elapsed time to resolve artifacts of all types in 1996 was more than three times as much as in 1995 (4300 elapsed hours per team vs. 1300 elapsed hours).

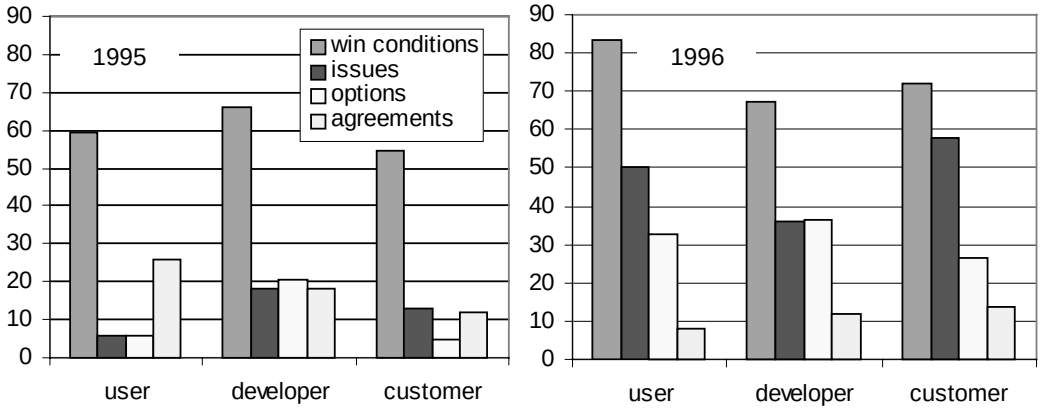


Figure 11: Average Time to Resolve Artifacts in Hours

KEY RESULTS DIFFERENCES

Both experiments exhibited a considerable number of commonalties and differences. In the following we will summarize them with special attention to the differences because they are a good source to reason about the effects of investigating hypothetical projects with more realistic ones:

1995 Projects

The 1995 project boundaries were well understood because of the limitation in choices. It was easier to make more complex dependencies in such an artificially constructed domain. Therefore, it is not surprising that it took less time to find issues and options. Furthermore, the 1995 projects were primarily constrained from a monetary point of view. This constraint may, however, have been the cause for longer resolution times of agreements since it affects the entirety (configuration) of the system, which is primarily captured through the agreements. This would also justify why the quantitative interdependencies of their artifacts were much more complex.

In 1995 the project setting did not require the students to go below high-level design issues. No implementation was required. Thus, it is not necessary to come up with a great number of negotiation items except for the ones, which are needed to solve the few built-in conflicts. Having fewer artifacts in a well understood domain would therefore reduce the time to resolve artifacts. Solutions in form of agreements were only needed to support high-level design issues and even there a number of design issues were already predefined and the students task was a matter of choosing between them

1996 Projects

The 1996 projects were only little understood by all participants (even the customers) because the problems they addressed constituted challenges which were unprecedented (e.g. web based technology was still new to many people). Also, the 1996 projects emphasized requirements completeness via the use of the domain taxonomy as a checklist. The need to understand the problem in more detail implies a higher demand in time. The extra investment of time is reflected in the higher number of artifacts (which is not caused by the higher number of participants as it was shown in [Boehm-Egyed, 1998]).

Due to some extent to the domain taxonomy stimulus (its structure and coverage), the 1996 projects exhibited less complex inter-artifact connectivity. Problems were divided into sub-problems and often resolved individually. This and the broader range of real library client desires and real COTS package constraints may have been responsible for increased time to resolve issues and options. It was harder to find options which would resolve issues than to finally vote on an agreement once something acceptable was found.

Since the 1996 negotiation guidelines were defined through its taxonomy, the negotiation was much more solution oriented by cross referencing the negotiation items to the table of contents of requirements documents. Thus a great number of agreements had to be created to cover all taxonomy items sufficiently.

CONCLUSIONS

Both projects also exhibited a number of commonalties. Sometimes these commonalties were weaker in one year than in the other, but they were still recognizable. The primary conclusions from the 1995 projects – that requirements negotiation patterns did not exhibit repeatability – was confirmed by the 1996 projects. The diversity in patterns of artifact creation and revision in Figure 8 and Figure 9 is roughly equivalent for the 1995 and 1996 projects (the greater uniformity for 1996 projects is due primarily to the shorter deadline for completing the negotiation). Thus, the goal of achieving repeatability in requirements engineering processes continues to appear unrealistic.

Nevertheless, we did observe some repeatability, most of which within a given year but some across the years. As such we found that the more realistic projects required more time to resolve conflicts and to identify options (alternatives) and that although more artifacts were created they exhibited less artifact interconnectivity. In terms of commonalties, we found that people factors such as experience did have effects onto negotiation patterns (especially in 1996) and that users and customers were most significant (in terms of artifact creation) during the goal identification whereas the developers were more significant in identifying issues and options. We also found that both years exhibited some strange although similar stakeholder participation patterns (e.g. see Figure 5). Thus, it is encouraging to know that at least some repeatability was accomplished – and that in an area where one would not have readily assumed their existence. Future analysis may shed additional light onto this issue.

ACKNOWLEDGEMENT

This research is sponsored by DARPA through Rome Laboratory under contract F30602-94-C-0195 and by the Affiliates of the USC Center for Software Engineering: Allied Signal, Bellcore, Boeing, Electronic Data Systems, Federal Aviation Administration, GDE Systems, Hughes Electronics, Institute for Defense Analysis, Litton Data Systems, Lockheed Martin, MCC, Motorola, Network Programs, Northrop Grumman, Rational Software, Raytheon, Science Applications International, Software Engineering Institute, Software Productivity Consortium, Sun Microsystems, TI, TRW, USAF Rome Laboratory, US Army Research Laboratory, and Xerox.

REFERENCES

Bate, R., et al., "A Systems Engineering Capability Maturity Model," Technical Report CMU/SEI-95-MM-003, Software Engineering Institute, Pittsburgh, PA 15213, November 1995.

Boehm, B.W. *Software Engineering Economics*, Prentice Hall, 1981.

Boehm, B.W. "A Spiral Model of Software Development and Enhancement," *IEEE Computer*, May 1988, pp. 61-72.

Boehm, B.W. and Ross, R. "Theory W Software Project Management: Principles and Examples," *IEEE Transactions on Software Engineering*, July 1989, pp.902-916.

Boehm, B.W., Bose, P., Horowitz, E., Lee, M.J. "Software Requirements As Negotiated Win Conditions", *Proceedings of ICRE*, April 1994, pp.74-83.

Boehm, B.W., "Integrated Software Engineering and System Engineering," *The Journal of NCOSE*, Vol. I, No. I, July/September 1994, pp. 61-67.

Boehm, B.W., Bose, P., Horowitz, E., Lee, M.J. “Software Requirements Negotiation and Renegotiation Aids: A Theory-W Based Spiral Approach”, *Proceedings of ICSE-17*, April 1995, pp.243-253.

Boehm, B.W., “Anchoring the Software Process,“ *IEEE Software*, July 1996, pp.73-82.

Boehm, B., Egyed, A., “WinWin Requirements Negotiation Processes: A Multi-Project Analysis,” Proceedings of the 5th International Conference on Software Processes, June 1998.

Boehm, B., Egyed, A., Kwan, J., Port, D., Shah, A., and Madachy, R., "Using the WinWin Spiral Model: A Case Study", IEEE Computer, July 1998, pp. 33-44.

Bullen, C.V., Bennet, J.L., “Learning from User Experience with Groupware”, Conference on Computer-Supported Cooperative Work, October 1990, pp.291-302.

Egyed, A., Boehm, B., “Analysis of System Requirement Negotiation Behavior Patterns,” *Proceedings of INCOSE-7*, August 1997, pp. 269-276.

Fisher, R., Ury W., “Getting to Yes,” *Penguin Books*, 1981.

Horowitz, E. “WinWin Reference Manual: A System for Collaboration and Negotiation of Requirements”, Center for Software Engineering, University of Southern California Technical Report, March 1996.

Humphrey W.S., *Managing the Software Process*, Addison-Wesley, 1989.

Lee, M.J., “Foundations of the WinWin Requirements Negotiation System,” Ph.D. Dissertation, Center for Software Engineering, University of Southern California Technical Report, May 1996.

Paulk, M.C., Weber, C.V., Curtis, B., Chrissis, M.B., *The Capability Maturity Model - Guidelines for Improving the Software Process*, Addison-Wesley, 1995.

Rechtin, E., Maier, M., *The Art of Systems Architecting*, CRC Press, 1997.

Rose, S., et al, “Integrated Systems and Software Engineering Process,” Software Productivity Consortium Report SPC-96001-CMC, Herndon, VA, May 1996.

BIOGRAPHIES

Alexander Egyed is a PhD student at the Center for Software Engineering at the University of Southern California. His research interests are in software architecture, design, and requirements elicitation. He received a Diplom-Ingenieur in Informatics from the Johannes Kepler University in Linz, Austria and a MS in Computer Science from the University of Southern California. He is a student member of the IEEE, ACM, and INCOSE.

Barry Boehm is the TRW Professor of Software Engineering and Director of the Center for Software Engineering at the University of Southern California. His current research involves the WinWin groupware system for software requirements negotiation, architecture-based models of software quality attributes, and the COCOMO II cost-estimation model. Boehm received a BA in mathematics from Harvard University and an MS and PhD in mathematics from the University of California at Los Angeles. He is an AIAA Fellow, an ACM Fellow, an IEEE Fellow, and a member of the National Academy of Engineering.