Spiral Lifecycle Increment Modeling for New Hybrid Processes

Raymond Madachy, Barry Boehm, and Jo Ann Lane

University of Southern California Center for Software Engineering,
941 W. 37th Place, Los Angeles, CA, USA
{madachy, boehm, jolane}@usc.edu

Abstract. The spiral lifecycle is being extended to address new challenges for Software-Intensive Systems of Systems (SISOS), such as coping with rapid change while simultaneously assuring high dependability. A hybrid plan-driven and agile process has been outlined to address these conflicting challenges with the need to rapidly field incremental capabilities. A system dynamics model has been developed to assess the incremental hybrid process and support project decision-making. It estimates cost and schedule for multiple increments of a hybrid process that uses three specialized teams. It considers changes due to external volatility and feedback from user-driven change requests, and dynamically re-estimates and allocates resources in response to the volatility. Deferral policies and team sizes can be experimented with, and it includes tradeoff functions between cost and the timing of changes within and across increments, length of deferral delays, and others. Both the hybrid process and simulation model are being evolved on a very large scale incremental project and other potential pilots.

1 Introduction

Our experiences in helping to define, acquire, develop, and assess 21st century SISOS have taught us that traditional acquisition and development processes do not work well on such systems [1][2]. We are using simulation modeling to help formulate and assess new processes to meet the challenges of these systems.

The systems face ever-increasing demands to provide safe, secure, and reliable systems; to provide competitive discriminators in the marketplace; to support the coordination of multi-cultural global enterprises; to enable rapid adaptation to change; and to help people cope with complex masses of data and information. These demands will cause major differences in the current processes [2].

We and others have been developing, applying, and evolving new processes to address SISOS. These include extensions to the risk-driven spiral model to cover broad (many systems), deep (many supplier levels), and long (many increments) acquisitions needing rapid fielding, high assurance, adaptability to high change traffic, and complex interactions with evolving Commercial Off-the-Shelf (COTS) products, legacy systems, and external systems.

The distinguishing features of a SOS are not only that it integrates multiple independently-developed systems, but also that it is very large, dynamically evolving, and

Q. Wang et al. (Eds.): SPW/ProSim 2006, LNCS 3966, pp. 167 – 177, 2006.

unprecedented, with emergent requirements and behaviors and complex socio-technical issues to address. Thus we have developed a system dynamics model because the methodology is well-suited to modeling these dynamic phenomena and their interactions [3].

1.2 The Scalable Spiral Model

The outlines of a hybrid plan-driven/agile process for developing a SISOS product architecture are emerging. It is a risk-driven balance of agility and discipline [4]. In order to keep SISOS developments from becoming destabilized from large amounts of change traffic, it is important to organize development into plan-driven increments in which the suppliers develop to interface specs that are kept stable by deferring changes, so that the systems can plug and play at the end of the increment. But for the next increment to hit the ground running, an extremely agile team needs to be concurrently doing continuous market, competition, and technology watch, change impact analysis, COTS refresh, and renegotiation of the next increment's prioritized content and the interfaces between the suppliers' next-increment interface specs.

The spiral model was introduced in 1986 and later elaborated for WinWin extensions [5]. It has continued to evolve to meet the needs of evolving development processes. We have been converging on a scalable spiral process model for SISOS that, for partial implementations to date, has scaled well from small e-services applications to super-large defense systems of systems, and multi-enterprise supply chain management systems.

Fig. 1 shows a single increment of the development and evolution portion of the model. It assumes that the organization has developed:

- A best-effort definition of the system's steady-state capability;
- An incremental sequence of prioritized capabilities culminating in the steady-state capability;
- A Feasibility Rationale providing sufficient evidence that the system architecture will support the incremental capabilities, that each increment can be developed within its available budget and schedule, and that the series of increments create a satisfactory return on investment for the organization and mutually satisfactory outcomes for the success-critical stakeholders.

As seen in Fig. 1, the model is organized to simultaneously address the conflicting challenges of rapid change and high assurance of dependability. It also addresses the need for rapid fielding of incremental capabilities with a minimum of rework, and the other trends involving integration of systems and software engineering, COTS components, legacy systems, globalization, and user value considerations.

The hybrid process uses a three-team cycle (lean, plan-driven, stabilized developers; thorough V&Vers; and agile, pro-active rebaseliners) that plays out from one increment to the next.

The need to deliver high-assurance incremental capabilities on short fixed schedules means that each increment needs to be kept as stable as possible. This is particularly the case for very large systems of systems with deep supplier hierarchies in

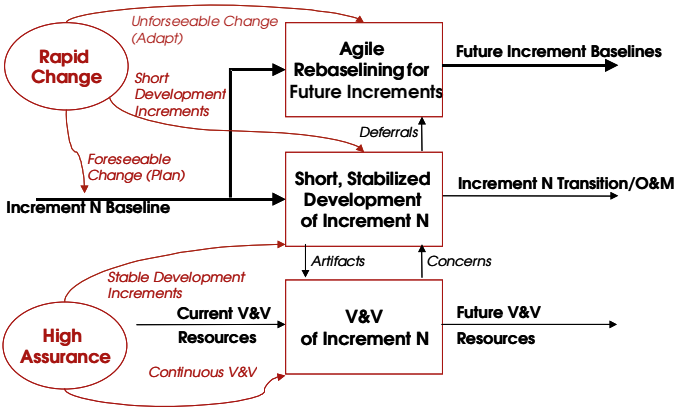

Fig. 1. The Scalable Spiral Process Model: Increment Activities

which a high level of rebaselining traffic can easily lead to chaos. The risks of destabilizing the development process make this portion of the project into a waterfall-like build-to-specification subset of the spiral model activities. The need for high assurance of each increment also makes it cost-effective to invest in a team of appropriately skilled personnel to continuously verify and validate the increment as it is being developed.

However, "deferring the change traffic" does not imply deferring its change impact analysis, change negotiation, and rebaselining until the beginning of the next increment. With a single development team and rapid rates of change, this would require a team optimized to develop to stable plans and specifications to spend much of the next increment's scarce calendar time performing tasks much better suited to agile teams.

The appropriate metaphor for addressing rapid change is not a build-to-specification metaphor or a purchasing-agent metaphor but an adaptive "command-control-intelligence-surveillance-reconnaissance" (C2ISR) metaphor. It involves an agile team performing the first three activities of the C2ISR "Observe, Orient, Decide, Act" (OODA) loop for the next increments, while the plan-driven development team is performing the "Act" activity for the current increment. "Observing" involves monitoring changes in relevant technology and COTS products, in the competitive marketplace, in external interoperating systems and in the environment; and monitoring progress on the current increment to identify slowdowns and likely scope deferrals. "Orienting" involves performing change impact analyses, risk analyses, and tradeoff analyses to assess candidate rebaselining options for the upcoming increments. "Deciding" involves stakeholder renegotiation of the content of upcoming increments, architecture rebaselining, and the degree of COTS upgrading to be done to prepare for the next increment. It also involves updating the future increments' Feasibility Rationales to ensure that their renegotiated scopes and solutions can be achieved within their budgets and schedules.

A successful rebaseline means that the plan-driven development team can hit the ground running at the beginning of the "Act" phase of developing the next increment, and the agile team can hit the ground running on rebaselining definitions of the increments beyond.

As much as possible, usage feedback from the previous increment is not allowed to destabilize the current increment, but is fed into the definition of the following increment. Of course, some level of mission-critical updates will need to be fed into the current increment, but only when the risk of not doing so is greater that the risk of destabilizing the current increment.

1.2 System Dynamics Modeling Introduction

System dynamics is a simulation methodology for modeling continuous systems. Quantities are expressed as levels, rates and information links representing feedback loops. It provides a rich and integrative framework for capturing myriad process phenomena and their relationships. System dynamics is well-suited to deal with the complexities of SOS because it captures dynamic feedback loops and interacting phenomena that cause real-world complexity [3].

Fig. 2 serves as a model diagram legend showing the notation for system dynamics elements in a simple system. These notations and following brief descriptions of the elements may help understand the model described in Section 2.

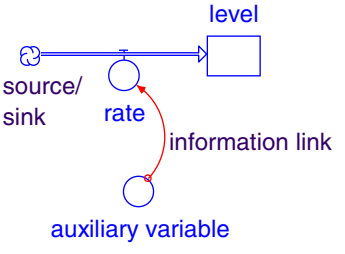

Fig. 2. System Dynamics Model Notation

Levels are the state variables representing system accumulations over time. They can serve as a storage device for material, energy, or information. Contents move through levels via inflow and outflow rates. Levels are a function of past accumulation of rates.

Sources and sinks represent levels or accumulations outside the boundary of the modeled system. Sources are infinite supplies of entities and sinks are repositories for entities leaving the model boundary.

Rates are also called flows; the "actions" in a system. They effect the changes in levels. Rates may represent decisions or policy statements. Rates are computed as a function of levels, constants and auxiliaries.

Auxiliaries are converters of input to output, and help elaborate the detail of stock and flow structures. An auxiliary variable must lie in an information link that connects a level to a rate. Auxiliaries often represent "score-keeping" variables.

Information links are used to represent information flow as opposed to material flow. Rates, as control mechanisms, often require links from other variables (usually levels or auxiliaries) for decision making. Information links can represent closed-path feedback loops between elements.

2 Model Overview

The primary portion of the system dynamics model diagram showing increment activities and the teams is in Fig. 3. It is built around a flow chain for capabilities and uses arrays to model multiple increments. The flow chains for the increment activities show multiple layers of levels and rates; these identify array elements that correspond to the increments. Thus the flow chain and its equations are arrays of five to model five increments (this preset number can be changed to model more or less increments).

Unanticipated changes arrive as a-periodic pulses via the *volatility trends* parameter. This is how they actually come on the projects vs. a constant level of volatility over time. The user can specify the pulses graphically (see the input for Volatility Profile in Fig. 4) or use formulas. The *capability volatility rate* will flow the changes into the corresponding increment for the current time.

From there they arrive in the level for *capability changes* and are then processed by the agile rebaselining team. They analyze the changes per the *average change analysis effort* parameter. Their overall productivity is a function of the *agile team size* (as specified by the user in Fig. 4) and the average analysis effort.

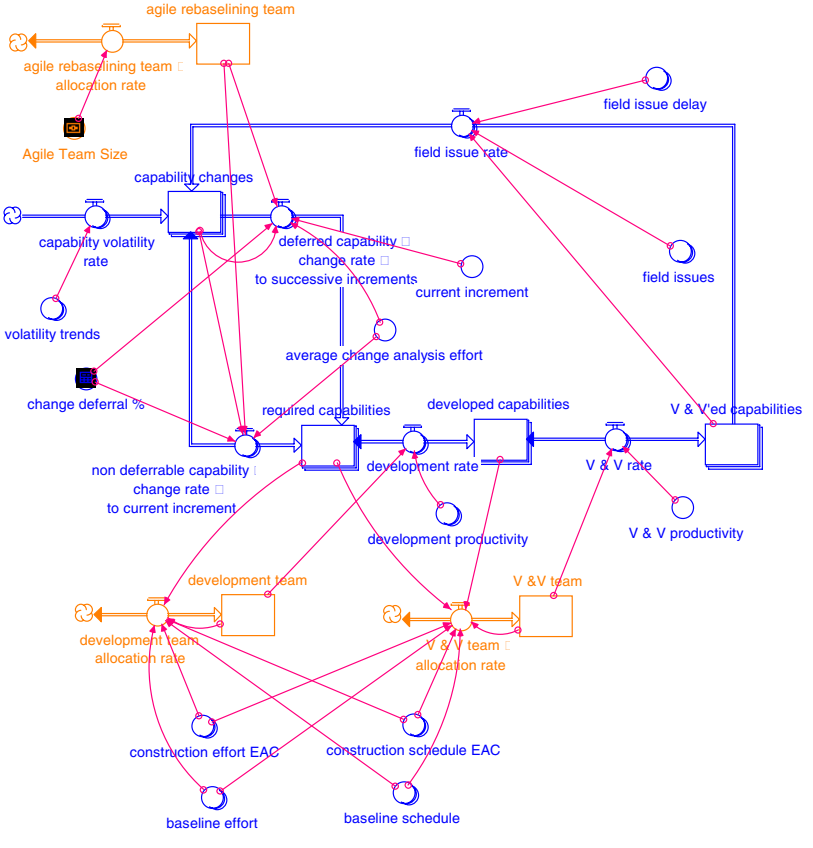

Fig. 3. Model Diagram

The *change deferral %* is a policy parameter to specify the percentage of changes that must be deferred to later increments via *deferred capability change rate to succeeding increments* to *required capabilities* for the appropriate increments. The remaining ones are non-deferrable that flow into the *required capabilities* for the current increment via the rate *non deferrable capability rate change to current increment*. The deferral policy parameter is also shown in the inputs in Fig. 4.

When an increment starts the *required capabilities* are developed by the development team at the *development rate* and flow into *developed capabilities* (all using the flow chain array index corresponding to the proper increment).

Similarly, the *developed capabilities* are then picked up the V&V team for their independent verification and validation. They do their assessment at the *V & V productivity rate* and the capabilities flow into *V & V'ed capabilities*.

The rates in the flow chain between *capability changes*, *required capabilities*, *developed capabilities* and *V & V'ed capabilities* are all bi-directional. This is a provision for capabilities to be "kicked back" or rejected by the various teams and sent back up the chain. For example, there are times when the developers have major concerns about a new capability and send it back to the re-baselining team. Likewise the V&V team might find some serious defects to be re-worked by the developers.

Finally there are user-driven changes based on field experience with the system. These are identified as *field issues* that flow back into the *capability changes* per the *field issue rate* at a constant *field issue delay* time. The *field issues* parameter represents the amount of concern with the fielded system and accounts for a primary feedback loop.

The agile baselining team is shown in the top left of the diagram. The size of the team can be specified as a constant size or a varying number of people over time via the inputs in Fig. 4. The *agile rebaselining team allocation rate* flows people in or out of the team to match the specified team size over time.

The development and V&V teams are shown at the bottom. Their allocation rates are based on the construction effort and schedule for the required capabilities known to-date. Currently the productivities and team sizes for development and V&V are calculated with a Dynamic COCOMO [6] variant. They are equivalent to COCOMO

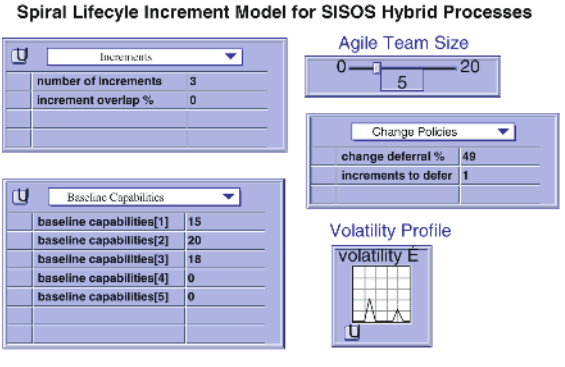

Fig. 4. Simulation Inputs

for a static project (the converse situation of this model context) and continuously re-calculated for changes. However, this aspect of the model whereby the team sizes are parametrically determined from size and effort multipliers will be refined so that constraints can be put on the development and V&V staff sizes.

An illustration of how the system responds to a volatility pulse in Increment 1 is in Fig. 5. In the figure legends, "[1]' refers to the increment number 1. An unanticipated set of changes occurs at month 8, shown as a *volatility trend* pulse. The changes immediately flow into the level for *capability changes*, which then starts declining to zero as an agile team of five people works it off per the average change analysis effort of four person-months.

The change is non-deferrable and it becomes incorporated into Increment 1, so the *total capabilities* for the increment increases. As the new capabilities become required for Increment 1, the development staffing responds to the increased scope by dynamically adjusting the team size to a new level.

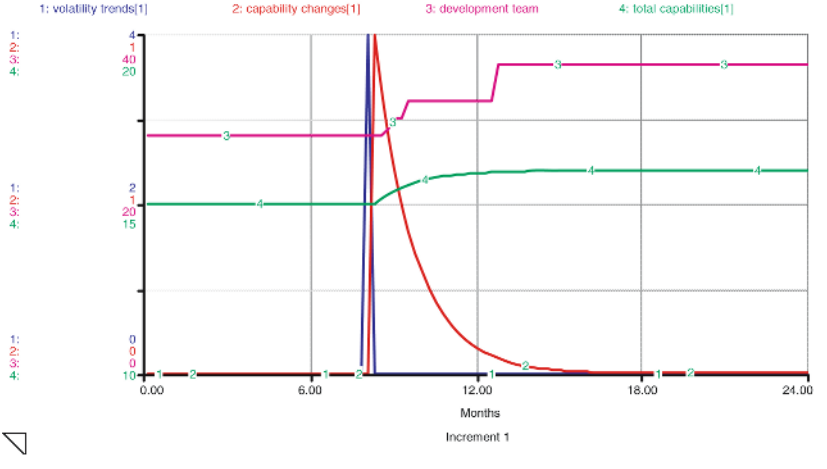

Fig. 5. System Response to Volatility – Increment 1

2.1 Tradeoff Functions

There are several functional relationships in the model that effect tradeoffs between deferral times and cost/schedule. For one, it is costlier to develop software when there is a lot of volatility during the development. If the required capabilities are added to an increment being developed, the overall effort increases due to the extra scope as well as the added volatility. The effort multiplier in Fig. 6 is used to calculate the construction effort and schedule based on a volatility ratio of total required capabilities to the baseline capabilities.

It is an aggregate multiplier for volatility from different sources. It works similarly to the platform volatility multiplier in COCOMO II [6], except in this context there may be many more sources of volatility (e.g. COTS, mission, etc.). This multiplier effect only holds for an increment when changes arrive midstream. If new changes are already in the required capabilities when an increment starts then it has no effect.

Additionally, the later a new capability comes in during construction the higher the cost to develop it. This is very similar to the cost-to-fix defects whereby the costs increases exponentially. Fig. 7 shows the lifecycle timing multiplier based on a ratio of the current time to the entire increment schedule.

Under normal circumstances, there is an additional cost of delaying capabilities to future increments because there is more of a software base to be dealt with and integrated into. Therefore we increase the cost of deferring to future increments by an additional 25% relative to the previous increment (this parameter is easily changed).

2.2 Dynamic Resource Allocation

In response to changes in the capabilities, the model calculates the personnel levels needed for the new increment size and interpolates for the amount of work done. If the increment has just started, then the interpolated staffing level will be closer to the higher level needed for the new Estimate-At-Completion (EAC). If the increment is mostly done, then it doesn't make sense to increase staff to the EAC level because almost all the work is done anyway.

A Rayleigh curve staffing version of the model intrinsically changes the staffing when changes occur with no interpolation necessary.

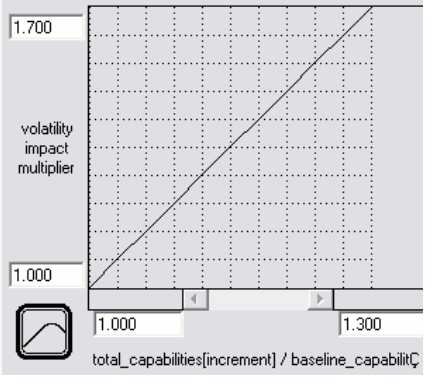

Fig. 6. Volatility Effort Multiplier

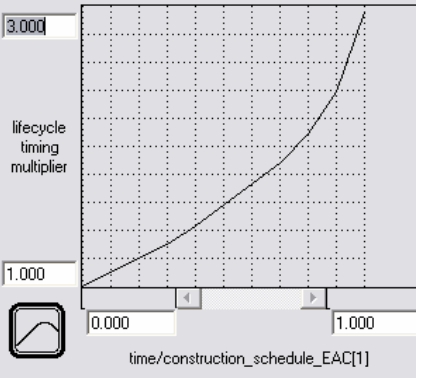

Fig. 7. Lifecycle Timing Effort Multiplier

2.3 Parameterizations

Since this is a macro model for very large systems, a capability is a "sky level" requirement measure. It is defined as a very high level requirement that we have made equivalent to 10 KSLOC for the purpose of estimation. The construction effort and schedule is currently calculated with a Dynamic COCOMO approach using the COCOMO II.2000 calibration [6].

The volatility impact multiplier is an extension of COCOMO for the SISOS situation. It is extrapolated from the current model and partially based on expert judgment. Other parameterizations relying on expert judgment include the average change analysis effort, lifecycle timing multiplier and amount of field issues. We are obtaining data on these and will be updating them based on the empirical data.

2.4 Sample Test Cases and Results

Table 1 shows the test cases results for varying the agile team size over two increments, each of 15 capabilities. The effort and schedule are for the development and V&V activities (the effort shown does not include the cost of the agile team, which does not account for substantial comparative differences). A change comes in at month eight (same as Fig. 5) and is processed by the agile team. The change is non-deferrable as it needs to be in Increment 1. However, the different team sizes will analyze the change at different rates.

The larger team size will process the change and incorporate it faster; hence the effort and schedule for Increment 1 improves with larger team size. However, if the team size is too small then it won't even make it into Increment 1. For team sizes of two and four it is processed too late and goes into Increment 2.

The total effort for four agile people is nearly equal to the total for a team size of ten (within 5%), since the change was effectively deferred and didn't incur lifecycle timing losses. However, the smaller team will also incur business value losses. These are not currently quantified in the model, but it is reasonable to assume that the value could far outweigh the 5% cost differential. Also not shown for the stretched out Increment 1 cases are losses due to late delivery.

Table 1. Test Case Results

	Increment 1		Increment 2		Total		Additional Losses
Agile Team Size (People)	Effort (PM)	Schedule (Mths.)	Effort (PM)	Schedule (Mths.)	Effort (PM)	Schedule (Mths.)	
2	728	32.3	2875	50.8	3603	83.1	Inc.1 business value
4	728	32.3	1171	37.8	1899	70.1	Inc.1 business value
6	1618	42	728	32.3	2346	74.3	
8	1448	40.5	728	32.3	2176	72.8	
10	1278	38.9	728	32.3	2006	71.2	

These results account for the lifecycle timing multiplier, volatility multiplier and increment delay losses. The model shows that a sufficient level of agile re-baseliners is necessary, or the cost and schedule for the project increases substantially. Enough must be on-board and productive enough to analyze the changes in a timely manner. Otherwise there could be a backlog of work to worry about at the beginning of a later increment that could have been resolved earlier by the agile team or other losses.

This set of test cases only varied agile team size, but another dimension to vary is the deferral percentage. Additionally we will simulate all five increments and have volatility occur in more than one increment in subsequent experiments.

3 Conclusions and Future Work

Processes need to rethought for current and upcoming SISOS, and the outlined hybrid process based on the scalable spiral model appears to be an attractive option. The dynamic model will help to further refine the hybrid process and determine optimized variants for different situations.

This first major iteration of the model already provides interesting results. It shows that if the agile team doesn't do their work, then developers will have to do it at a higher cost. Further experiments are underway to vary the deferral percentages, include rework, and constrain the staff sizes for development and V&V.

Both the hybrid process and the model will be further proven and evolved. Various improvements in the model are already identified and briefly discussed below, but further changes will come from users of the model. Additionally, empirical data to help calibrate and parameterize the model will come from users in the field and other data collection initiatives at USC.

This version of the model uses step function staffing profiles that adjust dynamically to changes. Another version uses Rayleigh curves for more realistic staffing patterns that adjust on the fly to midstream changes. These models will be integrated to allow the user to specify the type of staffing.

In the current test cases, only the optimum personnel levels are used for development and V&V, but in reality there may be staffing constraints. The model will be refined so users can constrain the development and V&V staff sizes. Another set of tests will compare tradeoffs between different agile team staffing policies (e.g. level-of-effort vs. demand-driven).

Patterns of changes and change policies will be experimented with. We will vary the volatility profiles across increments and demonstrate kick-back cases for capabilities flowing back up the chain from the developers or V&V'ers. Additionally we will model more flexible deferral policies across increments to replace the current binary simplification of allocating changes to the current or next increment.

As previously noted, the model currently does not account for business/mission value losses due to delays. Business value should be part of the overall process analysis, so provisions will be made to quantify the timed value of capabilities.

Parts of model have been parameterized based on actual empirical data, but not the change traffic. We will be getting actual data on volatility, change traffic trends and field issue rates from our USC affiliates and other users of the model. Data for the change analysis effort and volatility cost functions will also be analyzed.

After we get change data to populate the model and make other indicated improvements, we will be using it to assess increment risk for a very large scale SISOS program. It will also be used by contractors on the program in addition to our own independent usage to assess process options.

We also plan to apply it to other projects we are involved with, and the model will be provided to our USC-CSE industrial affiliates for assessing and improving their processes. They will also provide an opportunity to obtain additional empirical data for model parameters.

References

1. Boehm, B., Brown, A.W., Basili, V., Turner, R.: "Spiral Acquisition of Software-Intensive Systems of Systems", CrossTalk. May (2004)
2. Boehm, B.: "Some Future Trends and Implications for Systems and Software Engineering Processes", USC-CSE-TR-2005-507 (2005)
3. Madachy R.: Software Process Dynamics, IEEE Computer Society Press (2006)
4. Boehm, B., Turner, R.: Balancing Agility and Discipline, Addison Wesley (2003)
5. 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)
6. Boehm, B., Abts C., Brown A., Chulani S., Clark B.,Horowitz E., Madachy R.,Reifer D., Steece B.: Software Cost Estimation with COCOMO II, Prentice-Hall (2000)