The MBASE Life Cycle Architecture Milestone Package

No Architecture Is An Island

Barry Boehm, Dan Port, Alexander Egyed, and Marwan Abi-Antoun
Center for Software Engineering, University of Southern California, Los Angeles, CA 90089
{boehm, dport, aegyed, marwan}@sunset.usc.edu

Key words: Software architecture, systems architecting, architecture evaluation, model-based development, rationale capture.

Abstract: This paper summarizes the primary criteria for evaluating software/system architectures in terms of key system stakeholders' concerns. It describes the Model Based Architecting and Software Engineering (MBASE) approach for concurrent definition of a system's architecture, requirements, operational concept, prototypes, and life cycle plans. It summarizes our experiences in using and refining the MBASE approach on 31 digital library projects. It concludes that a Feasibility Rationale demonstrating consistency and feasibility of the various specifications and plans is an essential part of the architecture's definition, and presents the current MBASE annotated outline and guidelines for developing such a Feasibility Rationale.

1. ARCHITECTURE EVALUATION CRITERIA

A good software/system architecture satisfices among a number of potentially conflicting concerns. Table 1 (from Gacek et al., 1995), summarizes the major architecture-related concerns of a number of system stakeholders. These serve as a set of evaluation criteria for the architecture.

For example, the *customer* is likely to be concerned with getting first-order estimates of the cost, reliability, and maintainability of the software based on its high-level structure. This implies that the architecture should be strongly coupled with the requirements, indicating if it can meet them. The customer will also have longer-range concerns that the architecture be compatible with corporate software product line investments. *Users* need

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35563-4_35

P. Donohoe (ed.), *Software Architecture*

software architectures in order to be able to clarify and negotiate their requirements for the software being developed, especially with respect to future extensions to the product. The user will be interested at the architecting stage in the impact of the software structure on performance, usability, and compliance with other system attribute requirements. As with architectures of buildings, users also need to relate the architecture to their usage scenarios.

Table 1. Stakeholder concerns as architecture evaluation criteria.

Stakeholder	Concerns / Evaluation Criteria
Customer	• Schedule and budget estimation • Feasibility and risk assessment • Requirements traceability • Progress tracking • Product line compatibility
User	• Consistency with requirements and usage scenarios • Future requirement growth accommodation • Performance, reliability, interoperability, other quality attributes
Architect and System Engineer	• Product line compatibility • Requirements traceability • Support of tradeoff analyses • Completeness, consistency of architecture
Developer	• Sufficient detail for design and development • Framework for selecting / assembling components • Resolution of development risks • Product line compatibility
Interoperator	• Definition of interfaces with interoperator's system
Maintainer	• Guidance on software modification • Guidance on architecture evolution • Definition of interoperability with existing systems

Architects and systems engineers are concerned with translating requirements into high-level design. Therefore, their major concern is for consistency between the requirements and the architecture during the process of clarifying and negotiating the requirements of the system. *Developers* are concerned with getting an architectural specification that is sufficient in detail to satisfy the customer's requirements but not so constraining as to preclude equivalent but different approaches or technologies in the implementation. Developers then use the architecture as a reference for developing and assembling system components, and also use it to provide a compatibility check for reusing pre-existing components. *Interoperators* use the software architecture as a basis for understanding (and negotiating about) the product in order to keep it interoperable with existing systems. The *maintainer* will be concerned with how easy it will be to diagnose, extend or modify the software, given its high-level structure.

2. THE MBASE LIFE CYCLE APPROACH

In order to determine whether a software/system architecture is satisfactory, with respect to the criteria in Table 1, one needs considerably more than a specification of components, connectors, configurations and constraints. Considering the architecture as an island, entire of itself, puts one at a serious disadvantage in evaluating its adequacy.

We have been developing, applying and refining an approach called MBASE (Model-Based Architecting and Software Engineering) (Boehm-Port, 1998) to address this issue. It focuses on ensuring that a project's product models (architecture, requirements, code, etc.), process models (tasks, activities, milestones), property models (cost, schedule, performance, dependability), and success models (stakeholder win-win, IKIWISI (I'll Know It When I See It), business case) are consistent and mutually enforcing.

3. MBASE OVERVIEW

Figure 1 summarizes the overall framework used in the MBASE approach to ensure that a project's success, product, process and property models are consistent and well integrated. At the top of Figure 1 are various success models, whose priorities and consistency should be considered first. Thus, if the overriding top-priority success model is to "Demonstrate a competitive agent-based data mining system on the floor of COMDEX in 9 months," this constrains the ambition level of other success models (provably correct code, fully documented as a maintainer win condition). It also determines many aspects of the product model (architected to easily shed lower-priority features if necessary to meet schedule), the process model (design-to-schedule), and various property models (only portable and reliable enough to achieve a successful demonstration).

The achievability of the success model needs to be verified with respect to the other models. In the 9-month demonstration example, a cost-schedule estimation model would relate various product characteristics (sizing of components, reuse, product complexity), process characteristics (staff capabilities and experience, tool support, process maturity), and property characteristics (required reliability, cost constraints) to determine whether the product capabilities achievable in 9 months would be sufficiently competitive for the success models. Thus, as shown at the bottom of Figure 1, a cost and schedule property model would be used for the evaluation and analysis of the consistency of the system's product, process, and success models.

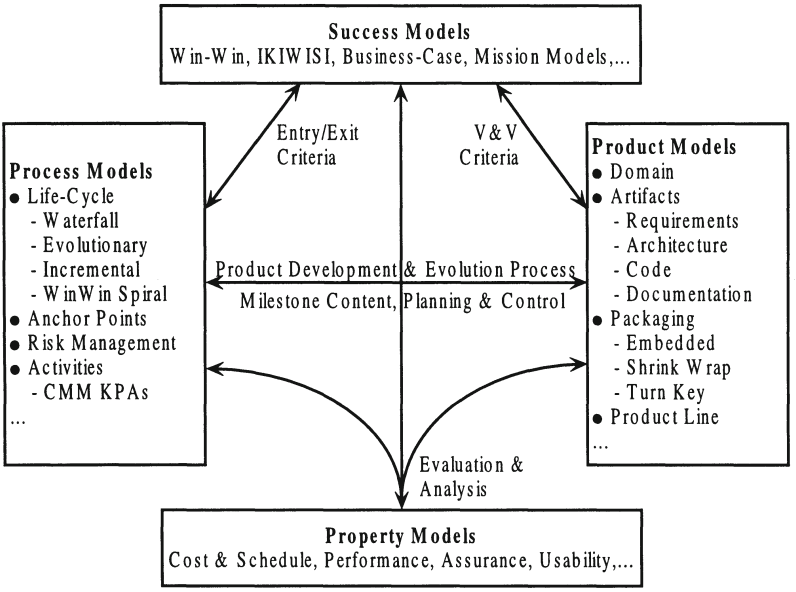


Figure 1. MBASE integration framework.

In other cases, the success model would make a process model or a product model the primary driver for model integration. An IKIWISI (I'll know it when I see it) success model would initially establish a prototyping and evolutionary development process model, with most of the product features and property levels left to be determined by the process. A success model focused on developing a product line of similar products would initially focus on product models (domain models, product line architectures), with process models and property models subsequently explored to perform a business-case analysis of the most appropriate breadth of the product line and the timing for introducing individual products.

3.1 Anchor point milestones

In each case, property models are invoked to help verify that the project's success models, product models, process models, and property levels or models are acceptably consistent. It has been found advisable to do this especially at two particular "anchor point" life cycle process milestones summarized in Table 2 (Boehm, 1996).

The first milestone is the Life Cycle Objectives (LCO) milestone, at which management verifies the basis for a business commitment to proceed at least through an architecting stage. This involves verifying that there is at

least one system architecture and choice of COTS/reuse components which is shown to be feasible to implement within budget and schedule constraints, to satisfy key stakeholder win conditions, and to generate a viable investment business case.

Table 2. Content of LCO and LCA packages.

Milestone Element	Life Cycle Objectives (LCO)	Life Cycle Architecture (LCA)
Definition of Operational Concept	• Top-level system objectives and scope – System boundary – Environment parameters and assumptions – Evolution parameters • Operational concept • Operations and maintenance scenarios and parameters • Organizational life-cycle responsibilities (stakeholders)	• Elaboration of system objectives and scope by increment • Elaboration of operational concept by increment
System Prototype(s)	• Exercise key usage scenarios • Resolve critical risks	• Exercise range of usage scenarios • Resolve major outstanding risks
Definition of System Requirements	• Top-level functions, interfaces, quality attribute levels, including: – Growth vectors – Priorities • Stakeholders' concurrence on essentials	• Elaboration of functions, interfaces, quality attributes by increment – Identification of TBDs (to-be-determined items) • Stakeholders' concurrence on their priority concerns
Definition of System and Software Architecture	• Top-level definition of at least one feasible architecture – Physical and logical elements and relationships – Choices of COTS and reusable software elements – Identification of infeasible architecture options	• Choice of architecture and elaboration by increment – Physical and logical components, connectors, configurations, constraints – COTS, reuse choices – Domain-architecture and architectural style choices – Architecture evolution parameters
Definition of Life-Cycle Plan	• Identification of life-cycle stakeholders – Users, customers, developers, maintainers, interfacers, general public, others • Identification of life-cycle process model – Top-level stages, increments – Top-level WWWWWHH* by stage	• Elaboration of WWWWWHH* for Initial Operational Capability (IOC) – Partial elaboration, identification of key TBDs for later increments
Feasibility Rationale	• Assurance of consistency among elements above – Via analysis, measurement, prototyping, simulation, etc. • Business case analysis for requirements, feasible architectures	• Assurance of consistency among elements above • All major risks resolved or covered by risk management plan

* WWWWWHH: Why, What, When, Who, Where, How, How Much.

The second milestone is the Life Cycle Architecture (LCA) milestone, at which management verifies the basis for a sound commitment to product development (a particular system architecture with specific COTS and reuse commitments which is shown to be feasible with respect to budget, schedule, requirements, operations concept and business case; identification and commitment of all key life-cycle stakeholders; and elimination of all critical risk items). The AT&T/Lucent Architecture Review Board technique (Marenzano, 1995) is an excellent management verification approach involving the LCO and LCA milestones. The LCO and LCA have also become key milestones in Rational's Objectory Process or Unified Management (Rational, 1997; Royce, 1998).

4. EXAMPLE MBASE APPLICATION

4.1 Digital library multimedia archive projects

Our first opportunity to apply the MBASE approach to a significant number of projects came in the fall of 1996. We arranged with the USC Library to develop the LCO and LCA packages for a set of 12 digital library multimedia applications. The work was done by 15 6-person teams of students in our graduate Software Engineering I class, with each student developing one of the 6 LCO and LCA package artefacts shown in Table 2. Three of the 12 applications were done by two teams each. The best 6 of the LCA packages were then carried to completion in our Spring 1997 Software Engineering II class.

Table 3. Example library multimedia problem statements.

Problem Set #2: Photographic Materials in Archives
Jean Crampon, Hancock Library of Biology and Oceanography
There is a substantial collection of photographs, slides, and films in some of the Library's archival collections. As an example of the type of materials available, I would like to suggest using the archival collections of the Hancock Library of Biology and Oceanography to see if better access could be designed. Material from this collection is used by both scholars on campus and worldwide. Most of the Hancock materials are still under copyright, but the copyright is owned by USC in most cases.
Problem Set #8: Medieval Manuscripts
Ruth Wallach, Reference Center, Doheny Memorial Library
I am interested in the problem of scanning medieval manuscripts in such a way that a researcher would be able to both read the content, but also study the scribe's hand, special markings, etc. A related issue is that of transmitting such images over the network.

Project Objectives
Create the artifacts necessary to establish a successful life cycle architecture and plan for adding a multimedia access capability to the USC Library Information System. These artifacts are:

1. An Operational Concept Definition
2. A System Requirements Definition
3. A System and Software Architecture Definition
4. A Prototype of Key System Features
5. A Life Cycle Plan
6. A Feasibility Rationale, assuring the consistency and feasibility of items 1-5

Team Structure
Each of the six team members will be responsible for developing the LCO and LCA versions of one of the six project artifacts. In addition, the team member responsible for the Feasibility Rationale will serve as Project Manager with the following primary responsibilities:

1. Ensuring consistency among the team members' artifacts (and documenting this in the Rationale).
2. Leading the team's development of plans for achieving the project results, and ensuring that project performance tracks the plans.

Project Approach
Each team will develop the project artifacts concurrently, using the WinWin Spiral approach defined in the paper "Anchoring the Software Process." There will be two critical project milestones: the Life Cycle Objectives (LCO) and Life Cycle Architecture (LCA) milestones summarized in Table 1.
The LCA package should be sufficiently complete to support development of an Initial Operational Capability (IOC) version of the planned multimedia access capability by a CS577b student team during the Spring 1997 semester. The Life Cycle Plan should establish the appropriate size and structure of such a team.

WinWin User Negotiations
Each team will work with a representative of a community of potential users of the multimedia capability (art, cinema, engineering, business, etc.) to determine that community's most significant multimedia access needs, and to reconcile these needs with a feasible implementation architecture and plan. The teams will accomplish this reconciliation by using the USC WinWin groupware support system for requirements negotiation. This system provides facilities for stakeholders to express their Win Conditions for the system; to define Issues dealing with conflicts among Win Conditions; to support Options for resolving the Issues; and to consummate Agreements to adopt mutually satisfactory (win-win) Options. There will be three stakeholder roles:

- Developer: The Architecture and Prototype team members will represent developer concerns, such as use of familiar packages, stability of requirements, availability of support tools, and technically challenging approaches.
- Customer: The Plan and Rationale team members will represent customer concerns, such as the need to develop an IOC in one semester, limited budgets for support tools, and low-risk technical approaches.
- User: The Operational Concept and Requirements team members will work with their designated user-community representative to represent user concerns, such as particular multimedia access features, fast response time, friendly user interface, high reliability, and flexibility of requirements.

Major Milestones

September 16, 1996	—	All teams formed
October 14, 1996	—	WinWin Negotiation Results
October 21-23, 1996	—	LCO Reviews
October 28, 1996	—	LCO Package Due
November 4, 1996	—	Feedback on LCO Package
December 6, 1996	—	LCA Package Due, Individual Critique Due

Individual Project Critique
The project critique is to be done by each individual student. It should be about 3-5 pages, and should answer the question, "If we were to do the project over again, how would we do it better - and how does that relate to the software engineering principles in the course?"

Figure 2. Multimedia archive project guidelines.

The multimedia archives covered such media as photographic images, medieval manuscripts, Web-based business information, student films and videos, video courseware, technical reports, and urban plans. The original Library client problem statements were quite terse, as indicated in Table 3. Our primary challenge was to provide a way for the student teams to work with these clients to go from these terse statements to an LCO package in 7 weeks and an LCA package in 11 weeks.

We enabled the students and clients to do this by providing them with a set of integrated MBASE models focused on the stakeholder win-win success model; the WinWin Spiral Model as process model; the LCO and LCA artifact specifications and a multimedia archive domain model as product models; and a property model focused on the milestones necessary for an 11-week schedule (see Figure 2). Further details are provided in (Boehm et al, 1997) and (Boehm et al, 1998).

4.2 MBASE model Integration for LCO stage

The integration of these models for the LCO stage is shown in Figure 3. The end point at the bottom of Figure 3 is determined by the anchor point postconditions or exit criteria for the LCO milestone (Boehm, 1996): having an LCO Rationale description which shows that for at least one architecture option, that a system built to that architecture would include the features in the prototype, support the concept of operation, satisfy the requirements, and be buildable within the budget and schedule in the plan.

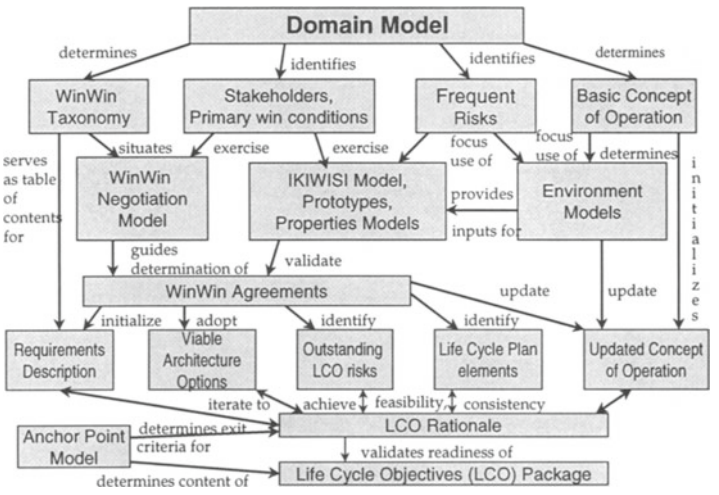
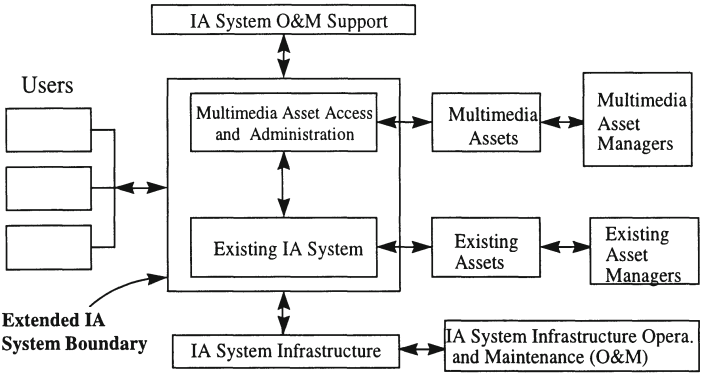


Figure 3. MBASE model integration: LCO Stage

The beginning point at the top of Figure 3 is the multimedia archive extension domain model furnished to the students, illustrated in Figure 4. The parts of the domain model shown in Figure 4 are the system boundary, its major interfaces, and the key stakeholders with their roles and responsibilities. The domain model also established a domain taxonomy used as a checklist and organizing structure for the WinWin requirements negotiation system furnished to the teams.

1. **System Block Diagram:**
   This diagram shows the usual block diagram for extensions providing access to and administration of multimedia information archive assets from an existing text-based information archive (IA) System:



   The system boundary focuses on the automated applications portion of the operation, and excludes such entities as users, operators, maintainers, assets, and infrastructure (campus networks, etc.) as part of the system environment. The diagram abstracts out such capabilities as asset catalogues and direct user access to O&M support and asset managers.

2. **Some Stakeholder Roles and Responsibilities**
   2.1 Asset Managers. Furnish and update asset content and catalogue descriptors. Ensure access to assets. Provide accessibility status information. Ensure asset-base recoverability. Support problem analysis, explanation, training, instrumentation, operations analysis.
   2.2-2.5 Similar roles and responsibilities defined for system operators, users, maintainers, and infrastructure service providers.

Figure 4. Multimedia archive extension domain model

As shown at the left of Figure 3, this taxonomy was also used as the table of contents for the requirements description, ensuring consistency and rapid transition from WinWin negotiation to requirements specification. The domain model also indicated the most frequent risks involved in multimedia archive applications. This was a specialization of the list of 10 most

frequent software risks in (Boehm, 1989), including performance risks for image and video distribution systems; and risks that users could not fully describe their win conditions, but would need prototypes (IKIWISI).

The sequence of activities between the beginning point and the LCO end point were determined by the WinWin Spiral Model. As illustrated in Figure 5, this model emphasizes stakeholder win-win negotiations to determine system objectives, constraints and alternatives; and early risk identification and resolution via prototypes and other methods (Boehm-Bose, 1994).

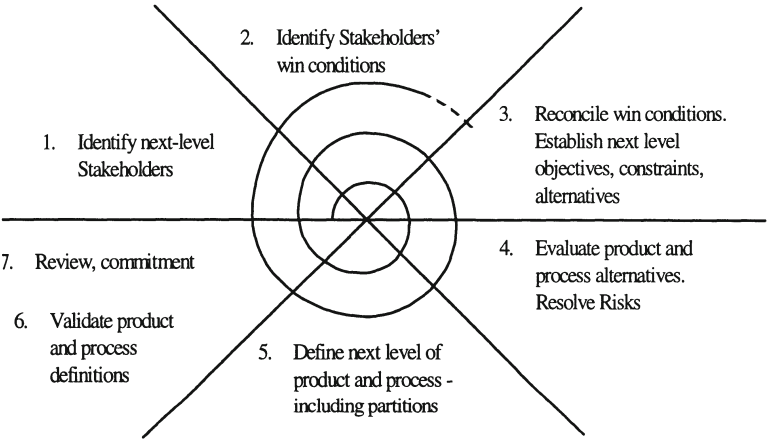


Figure 5. The WinWin spiral model

4.3 Project results

We were not sure how many of the 6-student teams would be able to work concurrently with each other and with their Library clients to create consistent and feasible LCO packages in 6 weeks and LCA packages in 11 weeks. With the aid of the integrated MBASE models, all 15 of the student teams were able to complete their LCO and LCA packages on time (3 of the applications were done separately by 2 teams). The Library clients were all highly satisfied, often commenting that the solutions went beyond their expectations. Using a similar MBASE and WinWin Spiral Model approach, 6 applications were selected and developed in 11 weeks in the Spring of 1997. Here also, the Library clients were delighted with the results, with one exception: an over-ambitious attempt to integrate the three photographic-image applications into a single product.

The projects were extensively instrumented, including the preparation of project evaluations by the librarians and the students. These have led to

several improvements in the MBASE model provided to the student teams for Fall 1997, in which 16 teams developed LCO and LCA packages for 15 more general digital library applications. For example, in 1996, the WinWin negotiations were done before the LCO milestone, while the prototypes were done after the LCO milestone. This led to considerable breakage in the features and user interface characteristics described in the LCO documents, once the clients exercised the prototypes. As a result, one of the top three items in the course critiques was to schedule the prototypes earlier. This was actually a model clash between a specification-oriented stakeholder win-win success model and the prototype-oriented IKIWISI success model. The 1997 MBASE approach removed this model clash by scheduling the initial prototypes to be done concurrently with the WinWin negotiations.

Another example was to remove several redundancies and overlaps from the document guidelines: as a result, the 1997 LCO packages averaged 110 pages as compared to 160 in 1996. The 1997 LCA packages averaged 154 pages as compared to 230 in 1996. A final example was to strongly couple the roles, responsibilities, and procedures material in the Operational Concept Description with the product transition planning, preparation, and execution activities performed during development. Further information on the 1997-98 projects is provided in (Boehm et al., 1998). 1996-97 and 1997-98 projects can be accessed via the USC-CSE web site at http://sunset.usc.edu/classes/classes.html.

5. THE ARCHITECTURE FEASIBILITY RATIONALE AS FIRST-CLASS CITIZEN.

As indicated in Table 2, the MBASE approach treats the Feasibility Rationale as a first-class citizen in the Life Cycle Objective and Life Cycle Architecture packages. For each of the LCO and LCA components in Figure 2, we have developed an annotated outline and set of guidelines for producing the component. Below is the current version for the Feasibility Rationale.

5.1 Document overview

Why (objective): The Feasibility Rationale (FR) is the glue that holds the other components of the Life Cycle Objective (LCO) and Life Cycle Architecture (LCA) packages together. It provides evidence of the feasibility and consistency of the LCO and LCA package components.

What (content): The Feasibility Rationale includes a business case analysis demonstrating that the resources invested in the project will

generate capabilities providing a satisfactory return on the investment. It also includes several satisfaction rationales addressing the various aspects of this question:

If I build the system using the given architecture and life cycle process, will it satisfy the requirements, support the operational concept, remain faithful to the key features determined by the prototype, and be achievable within the budgets and schedules in the life cycle plan?

Intended audience: The primary audiences are the LCO and LCA Architecture Review Boards. The parts dealing with client satisfaction must be understandable by the client representatives on the ARB. The technical parts must be sufficiently detailed and well-organized to enable the peers and technical experts to efficiently assess the adequacy of the technical rationale. The FR is also of considerable value to developers and other stakeholders in providing a rationale for key decisions made by the project.

Participants: The project manager is responsible for the overall content of the FR. Frequently, the business case is prepared by the author of the Operational Concept Description (OCD). Demonstrating the feasibility and consistency of portions of the LCO and LCA packages is the shared responsibility of the associated project participants. Other stakeholders may make their concurrence on win-win agreements contingent on demonstration of the agreement's feasibility in the Feasibility Rationale.

High level dependencies: The thoroughness of the Feasibility Rationale is dependent on the thoroughness of all the other LCO and LCA components. Issues incompletely covered in the Feasibility Rationale are a source of risk which should be covered in the Life Cycle Plan's (LCP) Risk Management section.

Overall tool support: Well-calibrated estimation models for cost, schedule, performance, or reliability are good sources of feasibility rationale. Others are prototypes, simulations, benchmarks, architecture analysis tools, and traceability tools (See Table 4 below for further information). The rationale capture capability in the WinWin tool is also useful.

5.2 Document outline

This section provides a table of contents for the Feasibility Rationale. Even though not all projects are alike, the people responsible for the Feasibility Rationale should consider all of these items carefully. If it is felt that some of them are not applicable, it should be noted as such for future reference. Similarly, the document outline can be expanded if there is a need. The recommended table of contents for the Feasibility Rationale document is as follows:

1. Overview
   1.1. Software Product Objectives
   1.2. Feasibility Rationale Objectives
2. Product Rationale
   2.1. Business Case Analysis
      2.1.1. Development Cost Estimate
      2.1.2. Operational Cost Estimate
      2.1.3. Estimate of Value Added and Relation to Cost
   2.2. Requirements Satisfaction
      2.2.1. Capability Requirements
      2.2.2. Interface Requirements
      2.2.3. Quality Requirements
      2.2.4. Evolution Requirements
   2.3. Operational Concept Satisfaction
   2.4. Stakeholder Concurrence
3. Process Rationale
   3.1. System Priorities
   3.2. Process Match to System Priorities
   3.3. Consistency of Priorities, Process and Resources

The following will explain in more detailed each of the items above, provide a rationale for them, show their dependencies to other sections within this document and to other documents, provide examples of their use, and give tool support recommendations whenever possible.

5.3 Document guidelines and rationale[1]

1. Overview
This section tells why the product and the plan are being developed.

1.1. Software Product Objectives
Provide a link to Section 1.1 of the Operational Concept Description (OCD). It contains a short description, in user terms, of the primary functions the product will perform, of its envisioned concept of operation, and of the user benefits expected from the product.

1.2. Feasibility Rationale Objectives
•To demonstrate that a system built using the specified architecture and life cycle process will satisfy the requirements, support the operational concept remain faithful to the key features determined by

[1] Text in **bold** can be used as is. Text in roman font indicates where project specific information needs to replace the general description provided. Text in *italic* font indicates specialization for Software Engineering I that would likely be tailored differently for other kinds of projects.

the prototype, and be achievable within the budgets and schedules in the life cycle plan.
•To rationalise development decisions in a way the prime audience (the customer and users) can understand
•To enable the customers to participate in the decision process and to express their satisfaction with the product

Integration and dependencies with other components:

- Item 1.1 is a link to the Objective items in Section 1.1 of the OCD.
- Item 1.2 may be used as is.

Additional guidelines:

None needed.

2. Product Rationale

This section furnishes the rationale for the product being able to satisfy the system specifications and stakeholders (e.g. customer, user).

2.1. Business Case Analysis

The Section describes the impact of the product in mainly monetary terms. How much does it cost to develop and to operate, how much added value does it generate, and thus how high is its return on investment. However, non-monetary factors may be also decisive. For instance, "added value" can include the improved quality of the service provided by the product.

2.1.1. Development Cost Estimate

Provide a summary of the full development cost, including hardware, software, people, and facilities costs.

2.1.2. Operational Cost Estimate

Provide a summary of the operational cost. Include also maintenance and administration cost and other costs which accumulate during transition of the product into production use (e.g. training).

2.1.3. Estimate of Value Added and Relation to Cost

Provide a summary of cost with and without the product and how much value is added by it. The value added may also describe non-monetary improvements (e.g. quality, response time, etc.) which can be critical in customer support and satisfaction. Include a return-on-investment analysis as appropriate.

2.2. Requirements Satisfaction

This section summarizes how well a system developed to the product architecture will satisfy the system requirements.

2.2.1. Capability Requirements
Show evidence that the system developed to the product architecture will satisfy the capability requirements, e.g., "capability described/demonstrated/exercised as part of included COTS component", with a pointer to the results. There is no need to restate obvious mappings from the requirements to the architecture.

2.2.2. Interface Requirements
Show evidence that the system developed to the product architecture will satisfy the interface requirements. These should include the interfaces and standards associated with the *University Computing Services* infrastructure and the *USC Integrated Library System.*

2.2.3. Quality Requirements
Show evidence that the system developed to the product architecture will satisfy the quality requirements.

2.2.4. Evolution Requirements
Show evidence that the system developed to the product architecture will satisfy the evolution requirements.

2.3. Operational Concept Satisfaction
Summarize product's ability to satisfy key operational concept elements, such as scenarios.

2.4. Stakeholder Concurrence
Summarize stakeholder concurrence by reference to WinWin negotiation results, memoranda of agreements, etc. Stakeholders may be anybody involved in the development process. For instance, a developer may claim that a certain response time cannot be achieved in a crisis mode unless nonessential message traffic is eliminated. Similarly, a customer may claim that the product does not satisfy his/her win conditions (e.g. cost). This section serves as a record of how such claims were resolved to the stakeholders' satisfaction.

Integration and dependencies with other components:

This section is highly dependent on all other documents. The cost estimates in Item 2.1 are strongly dependent on development cost (from LCP) and operational cost (from OCD). Item 2.2 maps requirements to design, which create a high dependency between the System and Software Requirements Description (SSRD), the System and Software Architecture Description (SSAD), and often the prototype. Similarly, item 2.3 creates a dependency between the OCD, the SSAD, and often the prototype. The stakeholder concurrence in Item 2.4 provides the basis for stakeholders to ratify their commitment to the project LCO and LCA packages at the ARB meetings.

Additional guidelines:

Table 4 summarizes the strengths and potential concerns for leading architecture attribute analysis methods. The rationale capture capability in the WinWin tool is also useful.

Table 4. Summary of software architecture attribute analysis methods

Method	Examples	Strengths	Potential Concerns
Current ADLs	RDD-100, StP, UML/Rose	• Static integrity (partial) • Traceability	• Dynamic integrity • Performance, cost, schedule analysis • Subjective attributes
New Generation ADLs	Rapide, Unicon, Wright	• Static, dynamic integrity • Some performance	• Model granularity and scalability • Cost, schedule, reliability, full performance • Subjective attributes
Scenario Analysis	SAAM	• Subjective attributes – Usability, Modifiability • Human-machine system attributes (partial) – Safety, security, survivability	• Largely manual, expertise-dependent • Scenario representativeness; method scalability • Verification/Validation/Accreditation • Integrity, performance, cost, schedule analysis
Simulation; Execution	Network 2.5; UNAS	• Performance Analysis • Some dynamic integrity • Some reliability, survivability	• Model granularity and scalability • Input scenario representativeness • Verification/Validation/Accreditation • Cost, schedule, subjective attributes
Parametric Modeling	COCOMO et al., Queuing Models, Reliability Block Diagrams	• Cost, schedule analysis • Reliability, availability analysis • Performance Analysis	• Subjective attributes • Static, dynamic integrity • Verification/Validation /Accreditation • Input validation

3. Process Rationale
This sections describes the rationale of the development process being able to satisfy the stakeholders (e.g. customer).

3.1. System Priorities
Summarize priorities of desired capabilities and constraints. Priorities may express time and date but also quality and others. (e.g. performance).

3.2. Process Match to System Priorities
Provide rationale for ability to meet milestones and choice of process model (e.g. anchor points in spiral model or increments, etc.).

3.3. Consistency of Priorities, Process and Resources
Provide evidence that priorities, process and resources match. E.g. budgeted cost and schedule are achievable; no single person is involved on two or more full-time tasks at any given time.

Integration and dependencies with other components:

Like the previous section, this section is also highly dependent on other documents, foremost the Life Cycle Plan (LCP) and System and Software Requirements Description (SSRD). Item 3.1 maps primarily to the capabilities in SSRD and milestones in LCP 2.2 and 2.3. Item 3.2 is a summary of LCP 4.2 which emphasis on priorities above. Item 3.3 is reasoning that the LCP is consistent and doable (especially LCP 4).

5.4 Potential pitfalls/best practices

The Feasibility Rationale is highly dependent on other components. Avoid duplicating these where mappings among components are obvious. In writing the Feasibility Rationale you should keep in mind that the primary audience is the Architecture Review Board (ARB), a mix of technical experts and general stakeholders. Portions of the FR should be tailored to the assessment needs of the various ARB members. Common pitfalls include over-reliance on vendor claims, neglect of critical off-nominal scenarios, and over-analysis of low-priority issues.

5.5 Quality criteria

The key quality criteria for the Feasibility Rationale are derived from its pitfalls. It needs to be highly consistent with the other components and it needs to be able to answer the key stakeholder questions about the feasibility of the product. It also needs to present selected system views demonstrating feasibility and consistency among the other components.

6. CONCLUSIONS

In specifying a software/system architecture, it is important not to treat the architecture as an isolated island. The architecture needs to be related to the operational concept it is supporting; the requirements the system will satisfy; the life cycle plan identifying the system's stakeholders, budgets and schedules; and any prototypes providing views of the desired system.

The satisfaction of these relationships is best recorded in a Feasibility Rationale for the architecture. For effective management review and commitment to the architecture, it is essential that the Feasibility Rationale be a first-class citizen in the architecture package. It is encouraging to note that this is so in the current draft of IEEE Standard 1471, "Recommended Practice for Architecture Description", (IEEE, 1998, Section 5.6)

REFERENCES

Boehm, B. (1989), *Software Risk Management*, IEEE-CS Press.

Boehm, B. (1996), "Anchoring the Software Process," *IEEE Software*, July, pp. 73-82.

Boehm, B. and Bose, P. (1994), "A Collaborative Spiral Process Model Based on Theory W," *Proceedings, ICSP3*, IEEE.

Boehm, B., Egyed, A., Kwan, J., and Madachy, R. (1997), "Developing Multimedia Applications with the WinWin Spiral Model," *Proceedings, ESEC/ FSE 97*, Springer Verlag.

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

Boehm, B. and Port, D. (1998), "Conceptual Modeling Challenges for Model Based Architecting and Software Engineering (MBASE)", Proceedings, 1997 Conceptual Modeling Symposium (P. Chen, ed.), Springer Verlag (to appear)

Gacek, C., Abd-Allah, A., Clark, B.K., and Boehm, B.W. (1995), "Focused Workshop on Software Architectures: Issue Paper," *Proceedings of the ICSE 17 Workshop on Software Architecture*, April.

Garlan, D., Allen, R., and Ockerbloom, J. (1995), "Architectural Mismatch: Why Reuse is So Hard," *IEEE Software*, November, pp. 17-26.

IEEE Architecture Working Group (1998), "IEEE Recommended Practice for Architectural Description, *IEEE Std 1471, Draft Version 3.0*, 3 July.

Kazman, R., Bass, L., Abowd, G., and Webb, M. (1994), "SAAM: A Method for Analyzing the Properties of Software Architectures," *Proceedings, ICSE 16,* ACM/IEEE, pp. 81-90.

Marenzano, J. (1995), "System Architecture Validation Review Findings," in D. Garlan, ed., *ICSE17 Architecture Workshop Proceedings*, CMU, Pittsburgh, PA.

Port, D. (1998), *Integrated Systems Development Methodology*, Telos Press (to appear).

Rational (1997) *Rational Objectory Process*, Version 4.1, Rational Software Corp., Santa Clara, CA.

Royce, W.E. (1998), *Unified Software Management*, Addison Wesley (to appear).