Feasibility Rationale Description (FRD)

A. Description

1. Purpose

The purpose of the Feasibility Rationale Description (FRD) is to ensure that the system developers have not just created a number of system definition elements, but have also demonstrated the feasibility and consistency of these elements. The FRD contents should:

- Ensure feasibility and consistency of the other system definition components (OCD, SSRD, SSAD, LCP, Prototype)
- Demonstrate a viable business case for the system
- Identify shortfalls in ensuring feasibility, consistency, and business case as project risk items for LCP
- Demonstrate that a system built using the specified architecture (described in the SSAD) and life cycle process (described in the LCP) will:
  - satisfy the requirements described in the SSRD
  - support the operational concept described in the OCD
  - satisfy the success-critical stakeholders in the OCD and LCP
  - remain faithful to the key features determined by the prototype described in the OCD and SSRD
  - stay within the budgets and schedules in the LCP
- Rationalize life cycle decisions in a way the prime audience (the customer and users) and other stakeholders can understand
- Enable the success-critical stakeholders to participate in the decision process and to record their satisfaction with the product and their commitment to fulfill their roles as defined in the LCP.

2. Completion Criteria

Below are the completion criteria for the Feasibility Rationale Description for the three major milestones:

- Life Cycle Objectives (end of Inception Phase)
- Life Cycle Architecture (end of Elaboration Phase)
- Initial Operational Capability (end of Construction Phase)

2.1 Life Cycle Objectives (LCO)

- Assurance of consistency among the system definition elements above for at least one feasible architecture (this should be done via appropriate combinations of analysis, measurement, prototyping, simulation, modeling, reference checking, etc.)
• Assurance of a viable business case analysis for the system as defined
• Assurance that all success-critical stakeholders are committed to support the project throughout its Elaboration phase

2.2 Life Cycle Architecture (LCA)
• Assurance of consistency among the system definition elements above for the architecture specified in the SSAD
• Assurance of a viable business case analysis for the system as defined
• Assurance that all success-critical stakeholders are committed to support the project throughout its life cycle
• Assurance that all major risks are either resolved or covered by a risk management plan

2.3 Initial Operational Capability (IOC)
• Feasibility rationale for future increments beyond IOC
• Validation of business case and Results Chain (OCD 2.1) assumptions

3. Intended Audience
• The primary audiences are the LCO and LCA Architecture Review Board members, and all TRR and RRR reviewers, including:
  • Key system stakeholders;
  • Experienced peers;
  • Technical Specialists in critical areas.
• 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 FRD is of considerable future value to developers and other stakeholders in providing a rationale for important decisions made by the project.

4. Participating Agents
• All stakeholders are responsible for consistency and feasibility of the system definition via their participation in Win-Win negotiations and ARB reviews.
• Agreements can be contingent on demonstration of feasibility or risk resolution.

5. Performing Agents
• The project manager is responsible for the FRD content.
• The OCD author is usually the best-prepared person to develop the business case
6. High Level Dependencies

The thoroughness of the Feasibility Rationale Description is dependent on the thoroughness of all the other system definition elements. Any issues incompletely covered in the Feasibility Rationale Description are sources of risk, whose management should be covered in the Life Cycle Plan’s (LCP) Risk Management and Monitoring Procedures section (LCP Error! Reference source not found.).

7. 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. The rationale capture capability in the Win–Win tool is also useful.

8. Degree of Detail and Tailoring

The degree of details of the FRD should be risk-driven (as with any MBASE model). If it’s risky to put an item in (e.g., unreliable cost or effort estimates, speculative value propositions), don’t put it in. If it’s risky not to put an item in (e.g., assessment or risks with potentially critical consequences), do put it in. Sections of the FRD may be tailored down, or consolidated for small or non-critical, well defined systems. Satisfaction of Capability and Interface requirements already documented in the SSAD should be referenced rather than repeated in FRD 2.2.3 and 2.2.4.

B. Document Sections

1. Introduction

1.1 Purpose of the Feasibility Rationale Description Document

- Summarize the purpose and contents of this document with respect to the particular project and people involved.

- Avoid generic introductions as much as possible: for instance, you can show how your particular Feasibility Rationale Description meets the completion criteria for the given phase.

Common Pitfalls:

- Simply repeating the purpose of the document from the guidelines

1.2 References

Provide complete citations to all documents, meetings and external tools referenced or used in the preparation of this document.

577 Guidelines:

A "complete citation" for CS577 should include the title of the document (in suitable bibliographic form), and with the explicit URL for the document. [This information is requested so that future researchers can find the cited document from an on-line archive.]
1.3 Change Summary

577 Guidelines:

For versions of the FRD since the last ARB, include a summary of changes made in the document to ease the review process.

2. Product Rationale

This section furnishes the rationale for the product being able to satisfy the system specifications and stakeholders (e.g. customer, user). It should also provide the rationale as to why the proposed system is better than the current system.

Integration and Dependencies with other components:

This section is highly dependent on all other documents. The cost estimates in FRD 2.1 are strongly dependent on development cost (from LCP) and operational cost (from OCD). FRD 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. It creates a dependency between the OCD, the SSAD, and often the prototype. The stakeholder concurrence in FRD 2.3 summarizes the findings so that a green light can be given to proceed with the development.

2.1 Business Case Analysis

This section analyses the product’s return on investment: \( \text{ROI} = \frac{(\text{Benefits} - \text{Costs})}{\text{Costs}} \).

- Costs Include development, transition, operations, and maintenance costs.
- Where possible benefits are expressed in financial terms compared to costs, such as increased sales and profits, or reduced operating costs.
- However, non-financial factors may be also decisive. For instance, “added value” can include the improved quality of the service provided by the product.
  - For a commercial system, the business case analysis will generally demonstrate an acceptable financial return on investment.
  - For a public service system, (health, education, defense, emergency services) the rationale would be expressed in terms of improvements in public service effectiveness as determined by analysis expressed by the users; or in terms of cost savings to achieve the desired level of effectiveness.

2.1.1 Development Cost Analysis

- Using estimates computed in the Budgets section (LCP Error! Reference source not found.), provide a summary of the full development cost, including hardware, software, people, and facilities costs.

Common Pitfalls:

- Repeating the analysis from LCP Error! Reference source not found.. Provide only a summary, and reference the detailed analysis.

2.1.2 Transition Cost Estimate

- Provide a rough estimate of costs to be incurred during the transition of the product into production.
Guidelines for MBASE

• These costs may include:
  • Training Time (trainer preparation and delivery time; trainee time)
  • Data preparation
  • COTS licenses
  • Operational readiness testing
  • Site preparation

Facilities preparation

Equipment purchase

2.1.3 Operational Cost Estimate

• Provide a summary of the operational costs, including license costs for COTS software, system administration costs, and database administration costs

2.1.4 Maintenance Cost Estimate

• Provide a summary of hardware and software maintenance costs if applicable. Life cycle cost models such as COCOMO II can be helpful.

Common Pitfalls:

• Repeating the analysis from LCP Error! Reference source not found. Provide only a summary, and reference the detailed analysis

2.1.5 Estimate of Value Added and Return on Investment

• Provide a summary of cost with and without the product, and how much value it adds

• The value added may also describe non-monetary improvements (e.g. quality, response time, etc.), which can be critical in customer support and satisfaction.

• Non-monetary, qualitative value added by the system may often be estimated by considering how well the proposed system helps achieve the Organization Goals (OCD 3.2). This is why measures for these goals can be important. Similarly for the Results Chain (OCD 2.1).

• Include a Return-On-Investment (ROI) analysis and a breakeven analysis as appropriate. A breakeven analysis takes the form of a graph that indicates the overall costs as surmised from FRD 2.1.1, 2.1.2, 2.1.3, and 2.1.4 (that is starting (up-front) costs plus cumulative costs) subtracted from the value added over time. Clearly indicate the point at which the overall cost is zero. This is where the project begins to show a positive return on the investment. If later cumulative costs potentially bring this return negative later (which is not uncommon, such as when there is an expected downstream cost to purchase a COTS product, upgrades, license fees, implement evolutionary requirements, anticipated domain changes, etc.), then be sure to extend the graph long enough to show this.

• [Consistent with Results Chain (OCD 2.1)]

• [Consistent with Results Chain (OCD 3.2)]
Common Pitfalls:

- Over focusing on costs and underemphasizing the value-added estimate and return on investment.

2.2 Requirements Satisfaction

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

Common Pitfalls:

- Simply restating the requirements, without showing how and why the proposed architecture guarantees that they will be met

- Requirements satisfaction is demonstrated by indicating explicitly (usually by model element trace references) why the SSRD requirements satisfy (are true for) the OCD operational concepts and then indicating why the SSAD designs satisfy (implement) the SSRD requirements. If this is done in the SSAD, it should be referenced rather than repeated in the FRD.

2.2.1 Operational Concept Satisfaction

- Summarize the product's ability to satisfy the key operational concept elements and critical scenarios, including critical off-nominal scenarios (failover or exception-handling Scenarios)

- Show explicitly why the SSRD requirements satisfy (are true for) the OCD operational concepts. Complete coverage of the System Capabilities (OCD 4.2) is essential. To a lesser degree, Project Goals and Constraints (OCD 4.2) and Levels of Service (OCD 4.4).

- [Consistent with Operational Scenarios (OCD 3.4.3), and all of OCD 4.]

2.2.2 Project Requirements Satisfaction

- Summarize how project requirements are being met through the approach adopted for the project and described in LCP 4.

- Explain explicitly (i.e. provide detailed references) how the designs and the plan for implementing the designs are compatible the Project Requirements (SSRD Error! Reference source not found.). Complete coverage of the Project Requirements is essential. Provide rationale only for complex or high-risk requirements. For most, a simple trace map or matrix from design elements to the requirements should suffice. For those covered in the LCP (e.g. choices of platforms, tools, or programming languages), a reference is sufficient.

- [Consistent with System Requirements (SSRD 2)]

2.2.3 Capability Requirements Satisfaction

- 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.

- Explain explicitly (i.e. provide detailed references) how the designs will implement the SSRD 3.3 Capability Requirements. Complete coverage of the Capability Requirements is essential. Provide rationale only for critical requirements. For most, a simple trace map or matrix from design elements to the requirements should suffice.
• There is no need to restate obvious mappings from the requirements to the architecture covered in the SSAD or by a requirements traceability tool. But for critical, higher-risk capabilities (e.g., natural language processing, pattern recognition, agent coordination) a full rationale with supporting evidence is needed.

• For each critical requirement, indicate:
  • Criticality: Describe how essential this requirement is to the overall system
  • Technical issues: Describe any design or implementation issues involved in satisfying this requirement. Provide evidence of their resolution via analysis, prototyping, simulation, or technology maturity studies.
  • Cost and schedule: Describe the relative or absolute costs associated with the technical issues associated with satisfying the requirement, and show their consistency with the budgets and schedules in the LCP.
  • Dependencies: Provide evidence of the feasibility of dependencies on COTS package capabilities, externally furnished components, etc.
  • Side effects: For solutions that may involve significant interactions with other requirements such as Levels of Service (see Table 13), provide evidence that the combination of requirements remains feasible.
  • Risks: Describe the circumstances under which this requirement might not be able to be satisfied, and what actions can be taken to reduce the probability of this occurrence.

• [Consistent with System Requirements (SSRD 3.2)]

### 2.2.4 Interface Requirements Satisfaction

• Show evidence that the system developed to the product architecture will satisfy the critical interface requirements.

• Explain explicitly (i.e. provide detailed references) which designs will implement the SSRD 4 Interface Requirements. Complete coverage of the Interface Requirements is essential. Provide rationale only for critical interfaces. For most, a simple trace map or matrix from design elements to the requirements should suffice.

• Potentially critical or high-risk interfaces include those with external developers; COTS or non-developmental items (NDI); separately evolving interfacing systems; complex user interfaces; and complex hardware interfaces.

• [Consistent with System Interface Requirements (SSRD Error! Reference source not found.)]

### 2.2.5 Level of Service Requirements Satisfaction

• Ambitious Level of Service requirements and their tradeoffs can be the most difficult requirements to satisfy, and the most difficult requirements to demonstrate satisfaction of in advance.

• Table 1 summarizes the most effective product and process strategies available for the major Level of Service attributes.

• Table 2 shows how some architecture/product strategies for ensuring a given Level of Service attribute reinforce or conflict with ensuring other attributes.

• Table 3 summarizes the most effective analysis methods available for given attributes and their tradeoffs with other attributes.

• Explain explicitly (through analysis, detailed references to prototypes, models, simulations, etc.) how the designs will satisfy the SSRD L.O.S. Requirements (SSRD 5). Complete coverage of the L.O.S.
Requirements is essential. Provide rationale only for critical requirements. For many, a simple trace map or matrix from design elements to the requirements should suffice.

- [Consistent with Level of Service Requirements (SSRD Error! Reference source not found.)]

**Common Pitfalls:**

- Not explicitly justifying why designs will satisfy L.O.S. Requirements. An unvalidated L.O.S. requirement is a serious project risk.

**Table 1: Level of Service Product and Process Strategies**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Product Strategies</th>
<th>Process Strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Interoperability</strong></td>
<td>Generality, Integrity Functions, Interface Specification, Layering, Modularity, Self-containedness</td>
<td>Interface Change Control, Interface Definition Tools, Interoperator Involvement, Specification Verification</td>
</tr>
<tr>
<td><strong>Usability</strong></td>
<td>Error-reducing User Input/output, Help/explanation, Modularity, Navigation, Parametrization, UI Consistency, UI Flexibility, Undo, User-programmability, User-tailoring</td>
<td>Prototyping, Usage Monitoring &amp; Analysis, User Engineering, User Interface Tools, User Involvement</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>Descoping, Domain Architecture-driven, Optimization (Code/ Algorithm), Platform-feature Exploitation</td>
<td>Benchmarking, Modeling, Performance Analysis, Prototyping, Simulation, Tuning, User Involvement</td>
</tr>
<tr>
<td><strong>Development Cost / Schedule</strong></td>
<td>Descoping, Domain Architecture-driven, Modularity, Reuse</td>
<td>Design To Cost/schedule, Early Error Elimination Tools And Techniques, Personnel/Management, Process Automation, Reuse-oriented Processes, User &amp; Customer Involvement</td>
</tr>
<tr>
<td><strong>Reusability</strong></td>
<td>Domain Architecture-driven, Portability Functions</td>
<td>Domain Architecting, Reuser Involvement, Reuse Vector Specification &amp; Verification</td>
</tr>
<tr>
<td><strong>All of Above</strong></td>
<td>Descoping, Domain Architecture-driven, Reuse (For Attributes Possessed By Reusable Assets)</td>
<td>Analysis, Continuous Process Improvement, Incentivization, Peer Reviews, Personnel/Management Focus, Planning Focus, Requirement/ design V&amp;V, Review Emphases, Tool Focus, Total Quality Management</td>
</tr>
<tr>
<td>Primary Attribute</td>
<td>Architecture Strategy</td>
<td>Other Attribute Reinforcement</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>Dependability</td>
<td>Input acceptability checking</td>
<td>Interoperability, Usability</td>
</tr>
<tr>
<td>Redundancy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Backup/recovery</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monitoring &amp; Control</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interoperability</td>
<td>Input acceptability checking</td>
<td>Dependability, Usability</td>
</tr>
<tr>
<td></td>
<td>Layering</td>
<td>Interoperability/ Portability, Reusability</td>
</tr>
<tr>
<td>Usability</td>
<td>Error-reducing user input/output</td>
<td>Dependability</td>
</tr>
<tr>
<td></td>
<td>Input acceptability checking</td>
<td>Dependability, Interoperability</td>
</tr>
<tr>
<td>Performance</td>
<td>Architecture balance</td>
<td>Cost/Schedule</td>
</tr>
<tr>
<td></td>
<td>Domain architecture-driven</td>
<td>Cost/Schedule</td>
</tr>
<tr>
<td>Evolvability/Portability</td>
<td>Layering</td>
<td>Interoperability, Reusability</td>
</tr>
<tr>
<td>Cost/Schedule</td>
<td>Architecture balance</td>
<td>Performance</td>
</tr>
<tr>
<td></td>
<td>Domain architecture-driven</td>
<td>Performance</td>
</tr>
<tr>
<td>Reusability</td>
<td>Domain architecture-driven</td>
<td>Interoperability, Reusability</td>
</tr>
<tr>
<td></td>
<td>Layering</td>
<td>Interoperability, Evolvability/ Portability</td>
</tr>
</tbody>
</table>
Table 3 Top-Level Field Guide to Software Architecture Attribute Analysis Methods

<table>
<thead>
<tr>
<th>Levels of Service</th>
<th>Methods</th>
<th>Examples</th>
<th>Potential Concerns</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Static integrity (partial)</td>
<td>Interface Checking</td>
<td>DOORS, RDD-100, Requisite Pro</td>
<td>• Dynamic integrity</td>
</tr>
<tr>
<td>• Traceability</td>
<td></td>
<td></td>
<td>• Performance, cost, schedule analysis</td>
</tr>
<tr>
<td>• Static, dynamic integrity</td>
<td>Formalized Models</td>
<td>Rapide, Wright, HDM, ACME, Prism</td>
<td>• Model granularity and scalability</td>
</tr>
<tr>
<td>• Security integrity</td>
<td></td>
<td></td>
<td>• Cost, schedule, reliability, full performance</td>
</tr>
<tr>
<td>• Interoperability</td>
<td></td>
<td></td>
<td>• Subjective attributes</td>
</tr>
<tr>
<td>• Subjective attributes: Usability, modifiability</td>
<td>Scenario Analysis</td>
<td>SAAM</td>
<td>• Largely manual, expertise-dependent</td>
</tr>
<tr>
<td>• Human-machine system attributes: Safety, security, survivability</td>
<td></td>
<td></td>
<td>• Scenario representativeness; method scalability</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Verification/Validation/Accreditation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Integrity, performance, cost, schedule analysis</td>
</tr>
<tr>
<td>• Performance analysis</td>
<td>Simulation; Execution</td>
<td>Network Simulators; Middleware Instrumentation</td>
<td>• Model granularity and scalability</td>
</tr>
<tr>
<td>• Dynamic integrity</td>
<td></td>
<td></td>
<td>• Input scenario representativeness</td>
</tr>
<tr>
<td>• Reliability, survivability, accuracy</td>
<td></td>
<td></td>
<td>• Verification/Validation/Accreditation</td>
</tr>
<tr>
<td>• Cost, schedule analysis</td>
<td>Parametric Modeling</td>
<td>COCOMO, SMERFS, Queuing Models</td>
<td>• Cost, schedule, subjective attributes</td>
</tr>
<tr>
<td>• Reliability, availability analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Performance analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.2.6 Evolution Requirements Satisfaction

- Show evidence that the system developed to the product architecture will satisfy the critical evolution requirements (e.g., show which parts of the architecture ensure an easy transition to support of lower-priority post-IOC features; to an upcoming COTS product replacement; to a significant increase in transaction volume; or to a combined operation with another application).

- [Consistent with Evolution Requirements (SSRD Error! Reference source not found.)]

2.3 Stakeholder Concurrence

- Summarize stakeholder concurrence by reference to:
  - Win–Win negotiation results
  - Memoranda of agreements
• Stakeholders may be anybody involved in the development process. For instance, a developer may non-concur because a certain response time cannot be achieved in a crisis mode unless nonessential message traffic is eliminated. Similarly, a customer may non-concur because the product does not satisfy his/her win conditions (e.g., fixed budget or schedule).

• This section serves as a record of how such claims were resolved to the stakeholders' satisfaction.

3. Process Rationale

This section analyzes the ability of the development to satisfy the stakeholders'(e.g. customer) cost and schedule constraints.

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). FRD 3.1 maps primarily to the capabilities in SSRD and milestones in LCP. FRD 3.2 is a summary of LCP, with emphasis on priorities above. FRD 3.3 is reasoning that the LCP is consistent and doable (especially LCP). FRD 3.4 is summarizing how such claims were resolved to the stakeholders' satisfaction.

3.1 System Priorities

• Summarize priorities of desired requirements. Capability and Interface Requirements may be expressed either in priority order or by priority category (e.g., Essential, Important, Desired, Optional or Very High...Very Low). Level of Service Requirements may be prioritized similarly or also by identifying Desired and Acceptable Levels of Service. Project and Evolution Requirements are a mix of Capability and Level of Service Requirements, and should be prioritized as such.

• These priorities should be consistent with the Organization Goals (OCD 3.2) and Project Goals (OCD 4.2) as well as System Requirements (SSRD 4.2).

• Priorities should be distributed somewhat evenly across the priority levels.

Common Pitfalls:

• Making everything an Essential or Very High priority.

• Prioritizing on a capability by capability basis instead of a requirement by requirement basis.

577 Guidelines:

• Use the Easy WinWin prioritization capability for multi-stakeholder prioritization.

3.2 Process Match to System Characteristics and Priorities

• Provide rationale for

  • Choice of process model: The decision table provides guidance on selecting an appropriate process model for various combinations of system objectives, constraints and alternatives.

  • Choice of increment, block, or build sequence in incremental development.
3.3 Consistency of Priorities, Process and Resources

- Provide evidence that priorities, process and resources match:
  - Budgeted cost and schedule are achievable
  - No single person is involved in two or more full-time tasks at any given time
  - Low priority features can be feasibly dropped to meet budget or schedule constraints
  - Using the estimated Effort (Person-months) and Schedule from Budgets (LCP Error! Reference source not found.), show that the staffing levels are enough, and that the project is achievable within the schedule.
  - It is important to use a credible and repeatable estimation technique for the Effort and the Schedule.

577 Guidelines:

- Given the team projects’ fixed 24-week schedule, use of the Schedule as Independent Variable (SAIV) process is highly recommended:
  a. Use the Easy WinWin prioritization capability to prioritize requirements
  b. Use COCOMO II and expert judgment to determine how many top-priority requirements can be developed with 90% confidence in 24 weeks. Define a useful Core Capability of such top-priority requirements as Increment 1.
  c. Architect the system for ease of adding and dropping borderline-priority features.
  d. Monitor and control progress and introduction of new higher-priority features to keep the Core Capability delivery feasible.
  e. If (usually) the Core Capability is finished early, add next-priority features as the schedule allows.
# Table 4: Process Model Decision Table

<table>
<thead>
<tr>
<th>Growth Envelope</th>
<th>Objectives, Constraints</th>
<th>Robustness</th>
<th>Alternatives Available Technology</th>
<th>Alternatives Architecture Understanding</th>
<th>Model</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limited</td>
<td></td>
<td></td>
<td>COTS</td>
<td>Buy COTS</td>
<td></td>
<td>Simple Inventory Control</td>
</tr>
<tr>
<td>Limited</td>
<td></td>
<td></td>
<td>4GL, Transform</td>
<td>Transform or Evolutionary Development</td>
<td></td>
<td>Small Business - DP Application</td>
</tr>
<tr>
<td>Limited</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Evolutionary Prototype</td>
<td></td>
<td>Advanced Pattern Recognition</td>
</tr>
<tr>
<td>Limited to Large</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>Waterfall</td>
<td></td>
<td>Rebuild of old system</td>
</tr>
<tr>
<td>Limited to Medium</td>
<td>Low-Medium</td>
<td>Low-Medium</td>
<td>High</td>
<td>Evolutionary Development</td>
<td></td>
<td>Data Exploitation</td>
</tr>
<tr>
<td>Limited to Large</td>
<td></td>
<td></td>
<td>Large Reusable Components</td>
<td>Medium to High Capabilities-to-Requirements</td>
<td></td>
<td>Electronic Publishing</td>
</tr>
<tr>
<td>Very Large</td>
<td>High</td>
<td></td>
<td></td>
<td>Risk Reduction &amp; Waterfall</td>
<td></td>
<td>Air Traffic Control</td>
</tr>
<tr>
<td>Medium to Large</td>
<td>Low</td>
<td>Medium</td>
<td>Partial COTS</td>
<td>Low to Medium</td>
<td></td>
<td>Software Support Environment</td>
</tr>
</tbody>
</table>

# Table 5: Conditions for Additional Complementary Process Model Options

<table>
<thead>
<tr>
<th>Design-to-cost or Design-to-schedule</th>
<th>Fixed Budget or Schedule Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incremental Development (only one condition is sufficient)</td>
<td>Fixed Budget or Schedule Available; Early Capability Needed; Limited Staff or Budget Available; Downstream Requirements Poorly Understood; High-Risk System Nucleus; Large to Very Large Application; Required Phasing With System Increments</td>
</tr>
</tbody>
</table>
4. Project Risk Assessment

Any combinations of capabilities or objectives whose feasibility is difficult to assure, are major sources of risk. Risk Assessment consists of risk identification, risk analysis and risk prioritization. Frequent major sources of risk and techniques for resolving them are given in Table 6. The project's overall life cycle strategy described in LCP 2.1 should be consistent with its approach to risk management. The initial set of risks defined here will be updated throughout the project.

- Identify the major sources of risk in the project.
- Provide a description of all identified risks for the project, including risk exposure quantities.
- For critical risks, indicate the following:
  - Description
  - Risk Exposure: Potential Magnitude and Probability of Loss
  - Risk Reduction Leverage: in reducing risk exposure
  - Actions to Mitigate Risk
  - Contingency Plan
- Identify low-priority requirements that can be left out in the case of schedule slippage

Table 6: Software Risk Management Techniques

<table>
<thead>
<tr>
<th>Source of Risk</th>
<th>Risk Management Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personnel shortfalls</td>
<td>Staffing with top talent; key personnel agreements; team-building; training; tailoring process to skill mix; peer reviews.</td>
</tr>
<tr>
<td>Schedules, budgets, process</td>
<td>Detailed, multi-source cost and schedule estimation; cost/schedule as independent variable; incremental development; software reuse; requirements descoping; adding more budget and schedule; outside reviews.</td>
</tr>
<tr>
<td>COTS, external components</td>
<td>Benchmarking; peer reviews; reference checking; compatibility prototyping and analysis; usability prototyping</td>
</tr>
<tr>
<td>Requirements mismatch</td>
<td>Requirements scrubbing; prototyping; cost-benefit analysis; design to cost; user surveys</td>
</tr>
<tr>
<td>User interface mismatch</td>
<td>Prototyping; scenarios; user characterization (functionality; style, workload); identifying the real users</td>
</tr>
<tr>
<td>Architecture, performance, quality</td>
<td>Simulation; benchmarking; modeling; prototyping; instrumentation; tuning</td>
</tr>
<tr>
<td>Requirements changes</td>
<td>High change threshold: information hiding; incremental development (defer changes to later increments)</td>
</tr>
<tr>
<td>Legacy software</td>
<td>Reengineering; code analysis; interviewing; wrappers; incremental deconstruction</td>
</tr>
<tr>
<td>Externally-performed tasks</td>
<td>Pre-award audits; award-fee contracts; competitive design or prototyping</td>
</tr>
<tr>
<td>Straining computer science</td>
<td>Technical analysis; cost-benefit analysis; prototyping; reference checking</td>
</tr>
</tbody>
</table>
**Guidelines for MBASE**

**Additional Guidelines:**

There are numerous risk identification and analysis tools that can be applied in this section (the COCOMO II Risk feature is useful here). However, they can only give guidelines, not real answers. The best preparation for this section is to try to construct the Feasibility Rationale Description and see where you have difficulties.

**Common Pitfalls:**

- Simply repeating the above table as your risk analysis
- Repeating your FRD 4 table in the Risk Management section of LCP (LCP 4.1)

**5. Analysis Results**

- Identify architectural alternatives and tradeoffs. Identify unfeasible architectures or rejected alternatives; document criteria for rejection to avoid having the rejected architectural alternative selected in ignorance at some other point.
- Describe feasible architectural alternatives which were rejected due to solution constraints on the way that the problem must be solved, such as a mandated technology. Those architectural alternatives may be reconsidered should the solution constraints be relaxed.

**5.1 Product Features**

**5.1.1 Advantages**

This paragraph shall provide a qualitative and quantitative summary of the advantages to be obtained from the new or modified system with respect to the Organization Goals and Activities. This summary shall include new capabilities, enhanced capabilities, and improved performance, as applicable, and their relationship to deficiencies identified in the Current System Shortfalls, as well as the rationale for new capabilities. For a quantitative analysis, you may reference the Business Case Analysis from the FRD 4.

You may also describe the relationship of this system with any other systems if they exist. Specify if this system is intended to be stand-alone, used as a component in a larger product, or one of a family of products in a product line. If the latter, this section discusses the relationship of this system to the larger product or to the product line.

**5.1.2 Limitations**

This paragraph shall provide a qualitative and quantitative summary of potential disadvantages or limitations of the new or modified system. These disadvantages and limitations shall include, as applicable, degraded or missing capabilities, degraded or less-than-desired performance, greater-than-desired use of computer hardware resources, undesirable operational impacts, conflicts with user assumptions, and other constraints. These are used either for stakeholder expectations management or as a basis for further negotiation of system capabilities or tradeoffs.

**5.1.3 Tradeoffs Considered**

This paragraph shall identify and describe major alternatives for the concept of operation of the system, their characteristics, the tradeoffs among them, and rationale for the decisions reached. Also discuss alternative architectures and their pros and cons.

**5.1.4 Changes Considered**

- These are changes considered but not included.
• In general, the results of the Win–Win requirements negotiation activity will be to drop or defer some capabilities from the initially proposed system. It is valuable to capture these for future reference, along with the rationale for dropping or deferring them. Some of those changes considered but not included may become Evolution Requirements.

• Include Reference to Win–Win artifact (if applicable)

• You may include a threshold for including some of the deferred capabilities (e.g., depending on the availability of a specific COTS package, etc.)

• [Consistent with Evolution Requirements (SSRD Error! Bookmark not defined. )]

5.2 Commercial-Off-The-Shelf and third-party solutions

• List the existing third-party (Commercial, Open Source, Legacy, and Reusable) components that have been and are investigated for use in the system.

• Reference any surveys or evaluations that have been done on these products.

• Consider whether there are products that must not be used, and state the reason.

• List the products in the following table:

<table>
<thead>
<tr>
<th>Component Name (Component X)</th>
<th>Status</th>
<th>Rationale for selection/rejection, and relevant references</th>
<th>Related components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement Name</td>
<td></td>
<td></td>
<td>Dependencies</td>
</tr>
</tbody>
</table>

• Selected
• Under evaluation
• Rejected
• Specify the rationale for selecting or rejecting the component.
• If the component is still under evaluation briefly specify what features are still being evaluated.
• List any references to completed or ongoing evaluations, and major information sources that have influenced the evaluation.

Products/Components required for setup or execution.

Products and components interacting with component X.

An example table is shown below:

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Status</th>
<th>Rationale for selection/rejection, and relevant references</th>
<th>Related components</th>
</tr>
</thead>
</table>
### Application Server (Req 02)

<table>
<thead>
<tr>
<th>Application Server (Req 02)</th>
<th>Dependencies</th>
<th>Interacting Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Tomcat 4.3</td>
<td>Application server that can manage about 1000 users per min <a href="http://tomcatreport.com">http://tomcatreport.com</a>, and the developers are familiar with Java</td>
<td>Java SDK 1.4.1, (Redhat Linux 9.0, Solaris 2.1, Windows 2003)</td>
</tr>
<tr>
<td>Microsoft IIS 5.0</td>
<td>Rejected</td>
<td>The application server seems to have problems with scalability (refer to COTS Assessment Report)</td>
</tr>
</tbody>
</table>

### Persistent data storage (Req 03)

<table>
<thead>
<tr>
<th>Persistent data storage (Req 03)</th>
<th>Dependencies</th>
<th>Interacting Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL ver 3.2</td>
<td>Under Evaluation</td>
<td>Considering the XML – Relational conversion capabilities of the database</td>
</tr>
<tr>
<td>MSSQL 2003</td>
<td>Under Evaluation</td>
<td>Considering the scalability of the database</td>
</tr>
<tr>
<td>Oracle 9i</td>
<td>Under Evaluation</td>
<td>Considering the amount of resources required to run the database.</td>
</tr>
</tbody>
</table>

### 6. Appendices

Provide details of cash flow and project earnings statement as appropriate.