EPRAM: Evolutionary Prototyping Risk Analysis & Mitigation (e-Commerce Software Development Process Document)

Annie I. Antón
Ryan A. Carter
Julia B. Earp
Laurie A. Williams

{anton, williams}@csc.ncsu.edu
Julia_Earp@ncsu.edu

Software Requirements Engineering Lab
North Carolina State University
Department of Computer Science
Technical Report #TR-2001-08

August 20, 2001

Abstract

E-commerce software developers are under pressure to develop applications at a record pace. The widespread lack of process discipline and procedural guidance available for such market driven environments highlights the need for a

framework to support practitioners' development efforts. We present a validated evolutionary software development process, the EPRAM (Evolutionary Prototyping with Risk Analysis and Mitigation) Model, for rapid development in e-commerce (electronic commerce) environments. The EPRAM Model explicitly addresses the challenges inherent in small-team rapid development projects requiring no more than 12 people. The model was intentionally designed to comply with the Level 2 KPAs (Key Process Areas) of the Software Engineering Institute's Capability Maturity Model (CMM); it combines evolutionary prototyping with an aggressive risk mitigation strategy to ensure that proposed requirements adhere to all established security and privacy policies. This process document details the model and includes a set of 12 document templates for practitioners to employ as needed in a rapid development effort.

Purpose of this Document

The evolutionary software engineering process is described in detail in this process document. This document provides a justification for using the evolutionary prototyping process (see glossary). It also recounts the events that should occur for software development, and it gives an ordering for such events.

Justification for the Evolutionary Process

There is a trend in the software engineering community to build systems using prototyping as the model on which to base development. Rising costs of software and the lower rates of systems meeting customer needs are helping to foster an interest in prototyping as the primary means of creating software. Reportedly, an evolutionary process helps in the following ways:

- Clarifies Management and User Requirements
  Due to constantly changing opinions, vague specifications, and indecisive clients, requirements may suffer from a lack of focus and direction. The direction in which the software should develop can be very unclear when software engineers employ traditional process methods. Evolutionary prototyping helps to alleviate this by embodying the better-understood requirements in a tangible form. Management and users can see their requirements in the prototyped systems, and therefore can validate the requirements reflected in the prototype [LSZ93]. By the iterative nature of the prototype's evolution, the customer gets the opportunity to accept, reject, or change the prototype, and in turn, accept, reject, or change the project's requirements.

- Uncovers Missing or Unknown Requirements
  Another benefit associated with evolutionary prototyping is that of discovering a more complete set of requirements for the software project. Once a prototype baseline is established, users can often find additional functions that the prototype must provide. In [Gra91], Graham presents this ideal facet of the evolutionary process by claiming that "the key is that prototypes are an excellent means of eliciting correct, consistent, and complete specifications of requirements." Through the use of evolutionary prototyping, developers are able to find out what requirements exist that have not been considered [Dav92].

- Allows for System Flexibility to Meet Changing Constraints
  Evolutionary prototyping is beneficial to practitioners who are struggling to build software systems that by their very nature are based on constantly shifting assumptions, constraints, and requirements. Lehman has summed up this problem by writing that software always encapsulates some assumptions. Some of these assumptions will be stated in the documentation. Some assumptions will be implicit. As software is being developed, the world on which the assumptions are based will change [Leh90]. Evolutionary development models seamlessly take these effects into account, allowing the project to adapt to the changes and uncertainties almost inevitably encountered in software system creation. In these models, commitments in the constraints and requirements are only held until the beginning of the next prototyping cycle, where they can changed and altered as needed. The evolutionary model's adaptability works to promote its acceptability within the software community. Developers working on particularly volatile projects often look first to evolutionary prototyping from which to base their project [NC98, FD89].

- Provides a Method Whereby Users, Management, and Developers Can Communicate about Systems
  Graham is a proponent of evolutionary prototyping for the purposes of facilitating communication between all stakeholders of the software project. The prototypes produced allow all involved to have a common 'language' from which to speak about the system. There is less room for argument about what a prototype does – unlike the misinterpretations stemming from the conflicting, ambiguous natural language specifications seen in the more traditional, non-evolutionary software development methods. Many stakeholders are also less intimidated by the presentation of a prototype than by the paper avalanche of specifications, designs, screen layouts, and diagrams found too often in methods such as the Waterfall model [Gra89, Gra91, Gra98]. Graham is not alone in his support of prototyping for communication [LSZ93].

- Allows Participants to Reflect on Lessons Learned During System Development.
  As opposed the linear Waterfall model (see glossary), traditional evolutionary prototyping allows developers to take what they have learned and experienced from the implementation of the earlier prototypes and correct their mistakes and build on the experiences to gradually refine the software system [Bea99]. This sentiment is shared

among software engineers, as there is a common belief that "evolutionary models of prototyping take advantage of knowledge acquired as development progresses [FD89]."

Thus, as we see above, evolutionary prototyping is good at finding missing, previously unknown requirements throughout development. Software processes with an evolutionary approach may help to lessen development time, but will always ease maintenance [Gra91]. Unlike popular linear models or some spiral models (see glossary) where early work products are paper at best, this model shows promise in that core systems evolve quickly and early in the development cycle. In the current online world where time is measured in days instead of months or even years, evolutionary prototyping may draw on its strengths to be the premiere process.

Recommendations

The process is recommended for a small team (approximately 4 to 6 people) working for approximately 800 development hours. It is appropriate for many development projects; yet this particular model was produced with an eCommerce (see glossary) application focus.

Roles, Groups, and Responsibilities

The process requests that nine roles be established for the project, where one person may have multiple roles or where multiple people can fulfill one role. In the same manner, these roles are assigned to one of three groups that collaborate in specific areas or for particular purposes during the project. Roles are listed below as lending themselves to individuals with either a technical or a management background. This allows the team members to make informed choices that will demonstrate their strengths throughout the project. The roles, groups, and suggested responsibilities are given as follows:

Role	Role Responsibilities	Type of Role	Group	Group Responsibilities
Project Manager (Technical)	1. Performs overall supervision of the project 2. Validates configuration management practices 3. Maintains project plan 4. Delegates requirements gathering 5. Performs implementation 6. Arbitrates problems as they occur	Technical	Software Engineering Group	1. Guides project direction 2. Aids in configuration management 3. Assists in project proposal 4. Leads documentation effort 5. Leads business-side efforts
Technical Writer	1. Oversees and maintains documentation 2. Takes meeting minutes and meeting documentation 3. Acts as the configuration management expert 4. Writes privacy policy and security policy and ensures that the work products adhere to these policies. 5. Performs implementation when necessary	Management or Technical		
Requirements Engineer	1. Leads requirements gathering efforts 2. Communicates with stakeholders, clients, end-users 3. Conducts interviews with stakeholders 4. Acts as interface to the customer 5. Documents requirements as well as policies elicited.	Management		
Business Manager	1. Develops business plan, if appropriate, or revenue plan. 2. Maintains business plan	Management		
Chief Architect	1. Oversees design of overall system 2. Categorizes implementation work 3. Does resource planning 4. Performs implementation	Technical	Software Quality Assurance Group	1. Develops tests for prototypes. Reviews code and documentation, including the project plan, requirements document, and design document
Quality Assurance Expert	1. Designs test cases 2. Executes tests 3. Maintains fault / bug reports 4. Performs implementation	Technical		
Lead Programmer	1. Delegates implementation work 2. Oversees defect resolution 3. Performs implementation	Technical	Software Development Group	1. Assists in user interface screen creation 2. Leads prototype development effort
Subject Specialist	1. Knows some special topic 2. Performs implementation	Management or Technical		
User Interface Design Specialist	1. Creates storyboard design (e.g. paper prototype or html editor generated design)	Management		

The Process – In General

The process is a tailored evolutionary prototyping-based process. Evolutionary prototyping is a form of software system creation in which developers gather the most complete, best-understood requirements possible; design and implement a prototype; and have the customer evaluate the prototype. The developers glean new requirements and clarify existing requirements based on this interaction. This process is repeated until a working system emerges that encompasses the true set of customer and system requirements.

The process allows for four development cycles. During the first cycle, some initial work is performed to establish the business case, plan the project, and to better understand the customer needs and wants, as well as to establish a basic understanding of what is required of the new software. The second cycle verifies that the team has understood the general direction to which the system needs to aspire. This cycle involves some throw-away (glossary) user interface prototypes created for the purpose of demonstrating to the customer the intentions of the group. The third cycle is the first round of evolutionary prototyping. In this phase, the first true embodiment of the requirements results in a working prototype. The customer evaluates this prototype and adds and modifies requirements. Finally, during the last development cycle, the prototype is evolved with the new and modified requirements. The system is tested and delivered to the customer.

During the Process

Specific areas must be addressed throughout the evolution of the software. One of the first items created, when appropriate or needed, is the business plan (glossary). Also in the initial development cycle, the project plan (glossary) is also written. Both plans must be maintained as the development progresses. The business plan helps to describe the project in the context of the market and must be modified to meet the true state of the project and market. In addition, as schedules, personnel, roles, responsibilities, project descriptions, or related documents are changed, created, added, or deleted during the course of the project, the project plan must be updated to reflect these modifications. The provided business plan and project plan templates can serve as guides as to what items must be included. The project manager with the assistance of the technical writer should be responsible for the maintenance of the project plan, while the business manager with the assistance of the technical writer is responsible for updating the business plan.

Privacy and security are aspects of eCommerce systems that often determine how usable any system will be. In recognition of the importance of these areas, privacy and security policies must be established from the beginning of the project. In the initial cycle, they are created – and to be effective, these policies must not only be maintained, but they must also be embodied in the system that is produced. To that end, it is the team's responsibility, under the guidance of the technical writer, that every document and prototype part that is produced adheres to the privacy and security policy.

In addition, configuration management (see glossary) must be performed throughout the project's lifetime. Configuration management is the responsibility of the technical writer and is validated by the project manager. It should be completed as set forth in the configuration management plan, which in turn is based, at least in part, on the provided configuration management template.

Finally, throughout the process, quality assurance (see glossary) is an important area on which to focus. All documents should be checked for errors and go through the process of being validated and verified. In addition, the prototypes themselves must be determined to be free of error and defects. The creation of tests and review of both documentation and code should be done continuously throughout system creation. The Quality Assurance Group leads this effort.

The Process – Cycle 1

Summary of what has been accomplished before the start of this cycle:
Nothing.

What this cycle accomplishes:
During the first cycle, initial customer input serves as the basis from with the business manager and the technical writer develop the [optional] business plan using the business plan template. Using this information, the project manager and technical writer establish the project plan with assistance from the entire group in accordance with the provided project plan template. This allows for the creation and understanding of roles, responsibilities, schedules,

and a project description. At approximately the same time, work should be performed to create the security and privacy policies that will guide the project. The team collaborates on the creation of the policies, and the policies and the adherence to these policies will serve as a measure of the quality of the system that is produced. This underscores the importance of security and privacy aspects of an eCommerce system.

After the policies are established, an initial set of requirements can be gathered from the customer, users, other software systems, and project team so as to contribute understanding to *what* the system must do. The requirements gathering effort is led by the requirements engineer under the direction of the project manager, who in turn delegates requirements gathering to the team. The requirements should be documented as set forth in the requirements document template with the technical writer leading the documentation creation effort. The cycle ends with a group review / risk (see glossary) meeting to analyze the group's efforts, directions, problems, and solutions. This meeting is performed as given in the risk / review meeting template.

Summary of what has been accomplished by the end of this cycle:

- Business Plan has been created using the Business Plan Template
- Project Plan has been created using the Project Plan Template
- Privacy and Security Policies have been created
- Initial Requirements Set has been documented using the Requirements Document Template
- Risk / Review Meeting has been held according to the Review and Risk Meeting Template
- Risks have been identified and corresponding mitigation strategies have been enacted as discussed in the Risk / Review Meeting

The Process – Cycle 2

Summary of what has been accomplished before the start of this cycle:

At the beginning of this cycle, the business plan has been developed, the project has been planned, the privacy and security policies exist, a core set of requirements has been discovered and documented, and risk mitigation strategies have been set into place.

What this cycle accomplishes:

To aid in understanding and to provide a more concrete communication path between the customer and the development team, the second cycle focuses on creating throwaway-user interface prototypes to ensure that development is proceeding in the right direction. The Software Development Group creates basic user interface screens with input and help from the entire team. These screens are archived by the technical writer according to the appropriate configuration management policies. In addition, and in parallel, the chief architect begins to work toward creating a system design based on the identified requirements.

The user interface screens are taken to the customer to bridge communication between the stakeholders. These screens serve as a tool to allow stakeholders to discuss the system and discover communication breakdowns and missing requirements. This process cycle also helps to ensure future development will follow the correct path. As the customer and developer(s) discuss the screens and the system, new or modified requirements may be discovered. Such requirements are documented in full by the technical writer as a complementary section to the original requirements document. The cycle ends with a risk / review meeting to provide a common understanding of the status of the project, as well as uncover and rank risks, create risk mitigation strategies, and judge the effectiveness of risk mitigation strategies. The system screens are archived and not used again.

Summary of what has been accomplished by the end of this cycle:

- User Interface Prototyping has been performed
- Initial System Design has begun using the Design Document Template
- Customer has evaluated the prototypes of the interface
- Requirement set has been modified according to the Requirements Document Template with the feedback from the interface evaluation
- Risk / Review Meeting has been held according to the Review and Risk Meeting Template
- Risks have been identified and corresponding mitigation strategies have been enacted as discussed in the Risk / Review Meeting

The Process – Cycle 3

Summary of what has been accomplished before the start of this cycle:
At the beginning of this point in the lifecycle, the requirements have been created and validated, system design has begun, and the customer has provided input into the concepts and initial screens of the software.

What this cycle accomplishes:
In accordance with the new requirements and customer's comments, the system design is modified and completed as the bridge between the existing set of requirements and prototype implementation. The chief architect is responsible for overseeing the design completion. As the system design takes form, the lead programmer delegates the implementation work that arises from the system design and together, the team codes. The quality assurance expert creates tests for the software components and system based off of the requirements. The system programming, debugging, and testing is performed until the requirements are met and the design is fulfilled.

When the prototype is completed, it is shown to the customer. The customer looks over the created system and along with the developers, may uncover requirements that are new or requirements that need to be changed. As before, the new or modified requirements are documented in full by the technical writer as a complementary section to the original requirements document. The cycle ends with a risk / review meeting to provide a common understanding of status of the project, as well as uncover and rank risks, create risk mitigation strategies, and judge the effectiveness of risk mitigation strategies.

Summary of what has been accomplished by the end of this cycle:
- The system design is completed according to the Design Document Template
- The first prototype that embodies the requirements and design has been created
- The first prototype has been shown to and evaluated by the customer
- New or modified requirements are identified and documented as provided for by the Requirements Document Template
- Risk / Review Meeting has been held according to the Review and Risk Meeting Template
- Risks have been identified and corresponding mitigation strategies have been enacted as discussed in the Risk / Review Meeting

The Process – Cycle 4

Summary of what has been accomplished before the start of this cycle:
At the beginning of this point in the lifecycle, the final set of requirements has been created and validated, and the customer has provided input into the first prototype.

What this cycle accomplishes:
In accordance with the new requirements and customer's comments, the system design is modified and completed as the bridge between the existing set of requirements and prototype implementation. The chief architect is responsible for overseeing the design modification. As the system design is reworked, the lead programmer delegates the implementation work that arises from the system redesign and together, the team codes. The quality assurance expert creates tests for the software components and system based off of the new requirements. The system programming and debugging is performed until the requirements are met and the design is fulfilled.

At the conclusion of system implementation, the developers thoroughly test the system, using the tests that have been created by the Software Quality Assurance Group and including any tests to complete testing coverage as necessary. As defects or bugs are found, the system programmer orchestrates the fixes needed to correct problems. After the system is tested, a review / risk meeting is held to wrap up system creation. Unresolved issues are discussed. Finally, the completed system is delivered to the customer. This system should encompass the needed functionality and the approved system qualities as dictated by the customer.

Summary of what has been accomplished by the end of this cycle:
- The system design has been updated in a form consistent with the Design Document Template
- The final prototype that embodies the requirements and design has been created
- The final prototype has gone through extensive testing as noted in the SQA Template
- Risk / Review Meeting has been held according to the Review and Risk Meeting Template
- Open issues have been identified and discussed
- The final prototype has been delivered to the customer

Areas of Special Concern

Just as there are reasons why an evolutionary prototyping base was chosen for this model, there are areas where practitioners using this model need to pay careful attention. If these drawbacks can be identified, they can be monitored to ensure the process results in a successful project. Such areas of special concern include:

- Difficulty of Control
  Software engineers who use evolutionary prototyping often face a process that seems difficult to manage and control. The prototyping literature demonstrates that management of evolutionary prototyping is a major contributor to the building of 'good' software. Control of this process lies in solid management practices, as it can be difficult for all stakeholders to agree on the direction and objectives of the software development. The contents of the next prototype may also be hard to determine as a number of revisions and decisions may need to be made by management. Finally, it is often hard to judge the level of completeness of the prototyped system as prototyped systems often lend themselves to being either over-evolved or delivered prematurely [BS96]. Various methods abound in handling the difficulty of control issue. Research has being done in the areas of limiting iteration cycles [Gra89, Gra91] and using risk analysis [BS96] for the purpose of controlling the evolutionary process.

- Functionality Creep Possibility
  A major benefit of evolutionary prototyping for a process model is that it leads to systems that meet known user requirements and yet can change to meet the user requirements that are discovered later. This helps to produce systems that the customers want and need. However, this too, can become burdensome as customers often try to continually request that new features be added – features that will add to development time and costs [Gra91, Gra98]. Stricter time limits and limits on the number of iteration cycles are sometimes used to mitigate this drawback.

- Poorly Structured Applications
  Traditionally, evolutionary prototypes have had a certain amount of poor structure associated with them. This structure deficiency is mainly the result of the iterative process of the evolutionary model [NC98]. Often, the structures that result from evolved prototypes are considered to be inefficient when compared to their linear system counterparts [BS96]. Care must be taken when cycling through prototype refinement processes so that the design is kept clean.

- Lack of Continued Commitment from Users
  End user involvement is a must-have for any evolutionary prototyping based project [LSZ93]. End users are an integral part of the process of refining requirements and evaluating prototypes. They must be involved for longer sustained periods than is traditionally required by other processes. Users have to be aware of the status of the prototypes and should know what to expect from the prototypes that they are shown [Bea99]. This is necessary as users sometimes view prototypes as the 'real' systems, even when the systems are not complete or fully specified [CL99].

In Conclusion

The process, as provided above, fulfills the tailored CMM level two: Repeatable level as set forth in the "Tailored CMM for a Small eCommerce Company" paper. The model itself may be tailored, as well, to fit the needs of the development group as long as the spirit is maintained.

References

[AE01a] A.I. Antón and J.B. Earp. Strategies for Developing Policies and Requirements for Secure Electronic Commerce Systems, To appear in *Recent Advances in Secure and Private E-Commerce*, Kluwer Academic Publishers, 2001.

[AE01b] A.I. Antón and J.B. Earp. The NCSU E-Commerce Studio: Supporting Multidisciplinary Project Driven Development for Secure Systems, Submitted to *Communications of the ACM*, 13 July 2001.

[BS96] Baskerville, R.L. and J. Stage. Controlling Prototype Development Through Risk Analysis. *MIS Quarterly*, 20(4), Dec., pp. 481-504, 1996.

[Bea99] Beaumont, R. *Information Systems (IS) Development Methods*. Source: Laptop; C:\HIcourseweb new\chap12\s3\des1.doc, 1999. Online: http://www.robinbt2.free-online.co.uk/virtualclassroom/chap12/s3/des1.htm

[CAD01] R.A. Carter, A.I. Antón, A.Dagnino and L. Williams. Evolving Beyond Requirements Creep: A Risk-Based Evolutionary Prototyping Model. To appear *5th IEEE International Symposium on Requirements Engineering*, Toronto, Canada, August 2001.

[CL99] Constantine L. and L. Lockwood. *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. Addison-Wesley, 1999.

[Cul94] Culver-Lozo, K. Rapid Iteration in Software Process Improvement: Experience Report. 'Applying the Software Process', *Proceedings of the Third International Conference on the Software Process*, pp. 79-84, 1994.

[Dav92] Davis, A.M. Operational Prototyping: A New Development Approach. *IEEE Software*, Volume: 9 Issue: 5, Sept., pp. 70-78, 1992.

[Gra89] Graham, I. Structured Prototyping for Requirements Specification of Expert Systems. *IEEE Colloquium on Expert Systems Lifecycle*, pp. 5/1-5/3, 1989.

[Gra91] Graham, I. Structured Prototyping for Requirements Specification in Expert Systems and Conventional IT Projects. *Computing & Control Engineering Journal*, Volume: 2 Issue: 2, March, pp. 82-89, 1991.

[Gra98] Graham, I. *Requirements Engineering and Rapid Development – An Object-Oriented Approach*. Addison-Wesley, 1998.

[Leh90] Lehman, M.M. The Role of Process Models in Software and Systems Development and Evolution. 'Experience with Software Process Models', *Proceedings of the 5th International Software Process Workshop*, pp. 91-93, 1990.

[LSZ93] Lichter, H., M. Schneider-Hufschmidt, and H. Zullighoven. Prototyping in Industrial Software Projects – Bridging the Gap Between Theory and Practice. *Proceedings of the 15th International Conference on Software Engineering*, May, pp. 221-229, 1993.

[NC98] Nunes, N. and J.F. Cunha. Case Study: SITINA - A Software Engineering Project Using Evolutionary Prototyping. *Proc. CAiSE'98/IFIP 8.1 EMMSAD'98 Workshop*, 1998.

Process Diagram for the Evolutionary Process Model

Software Engineering Seminar Group

North Carolina State University
Department of Computer Science

Fall 2000

Key for Process Diagram

Activity that begins at one specific time and is performed continuously throughout the remainder of the process

Activity that is performed throughout the entire process

Activity that is performed at one given time in the process

Work product or document that is produced

Prototype that is produced

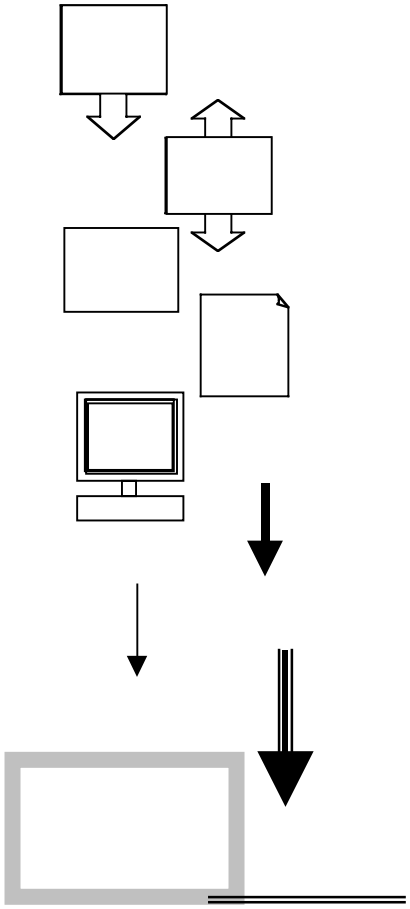
Major (control) flow through the process

Minor (data) flow through the process

Risk Mitigation flow

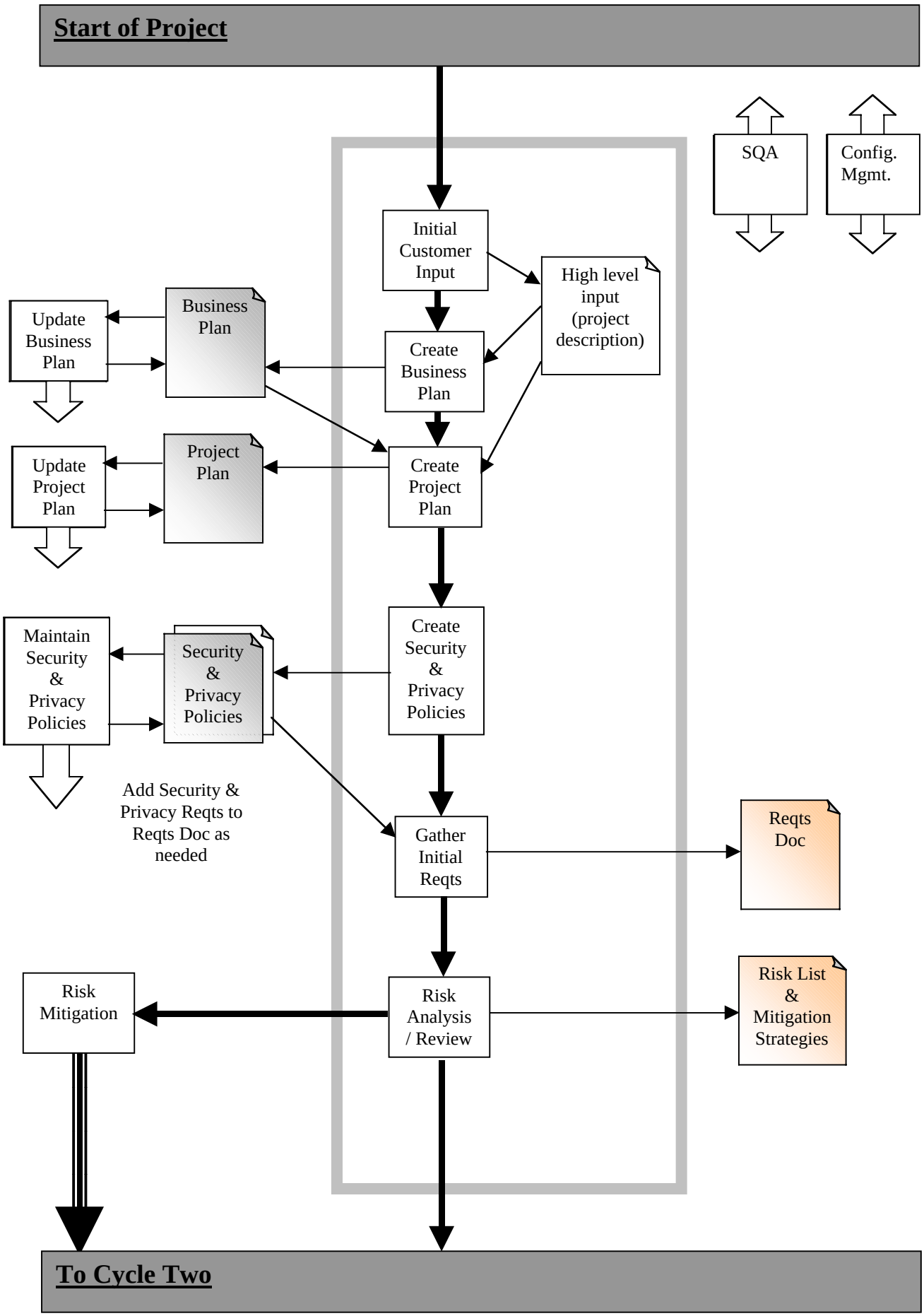
Cycle Boundary

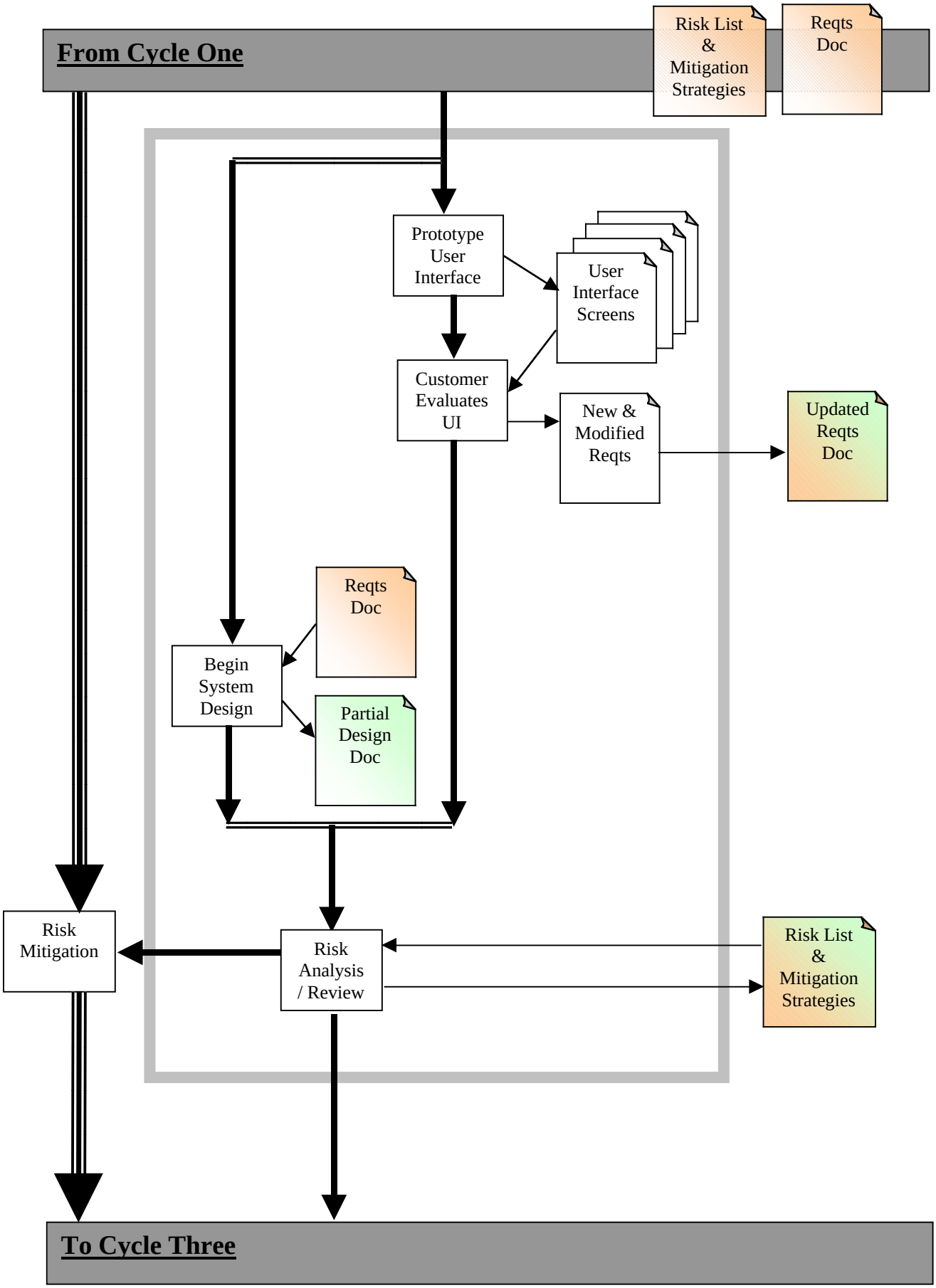
Flow Separation / Join

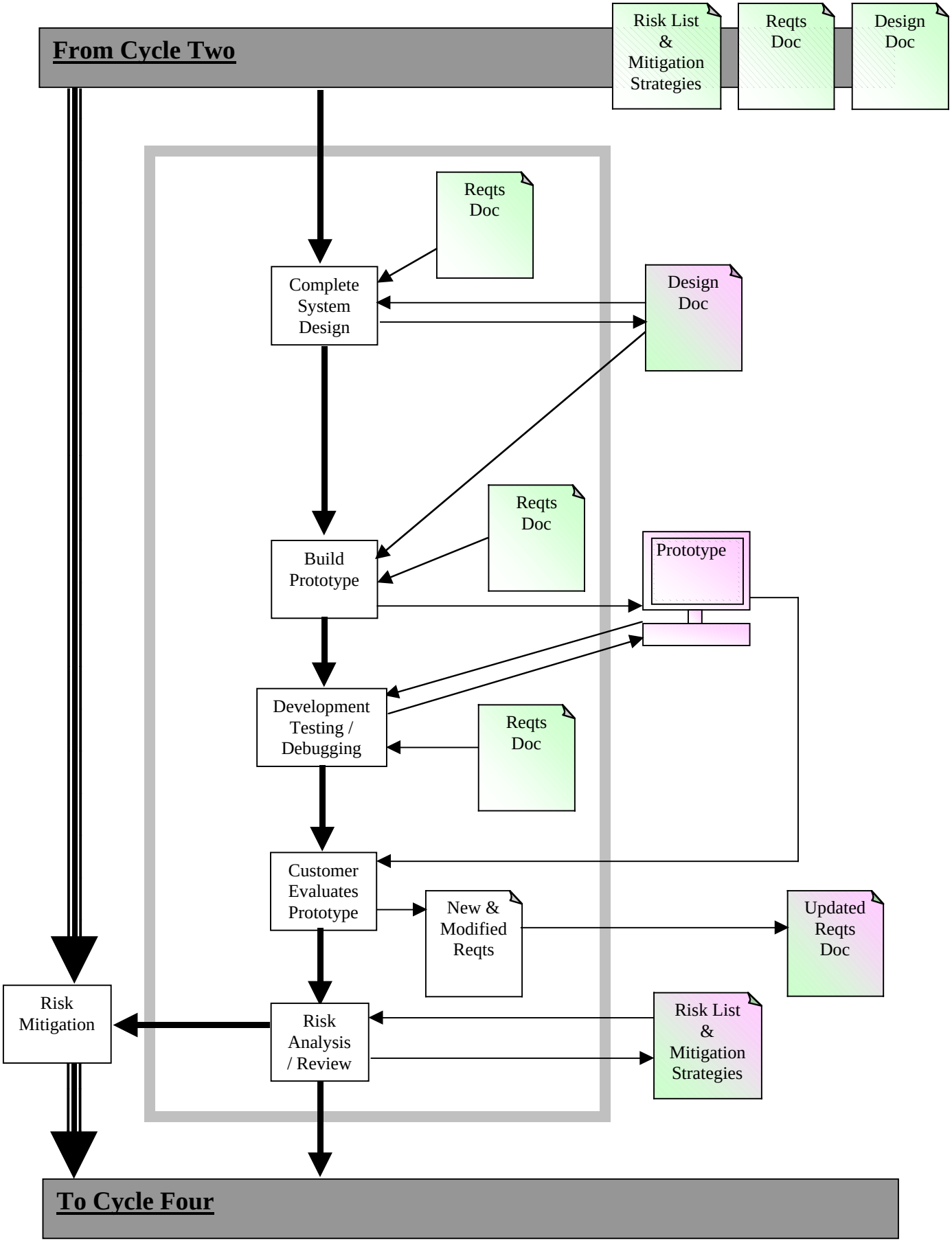


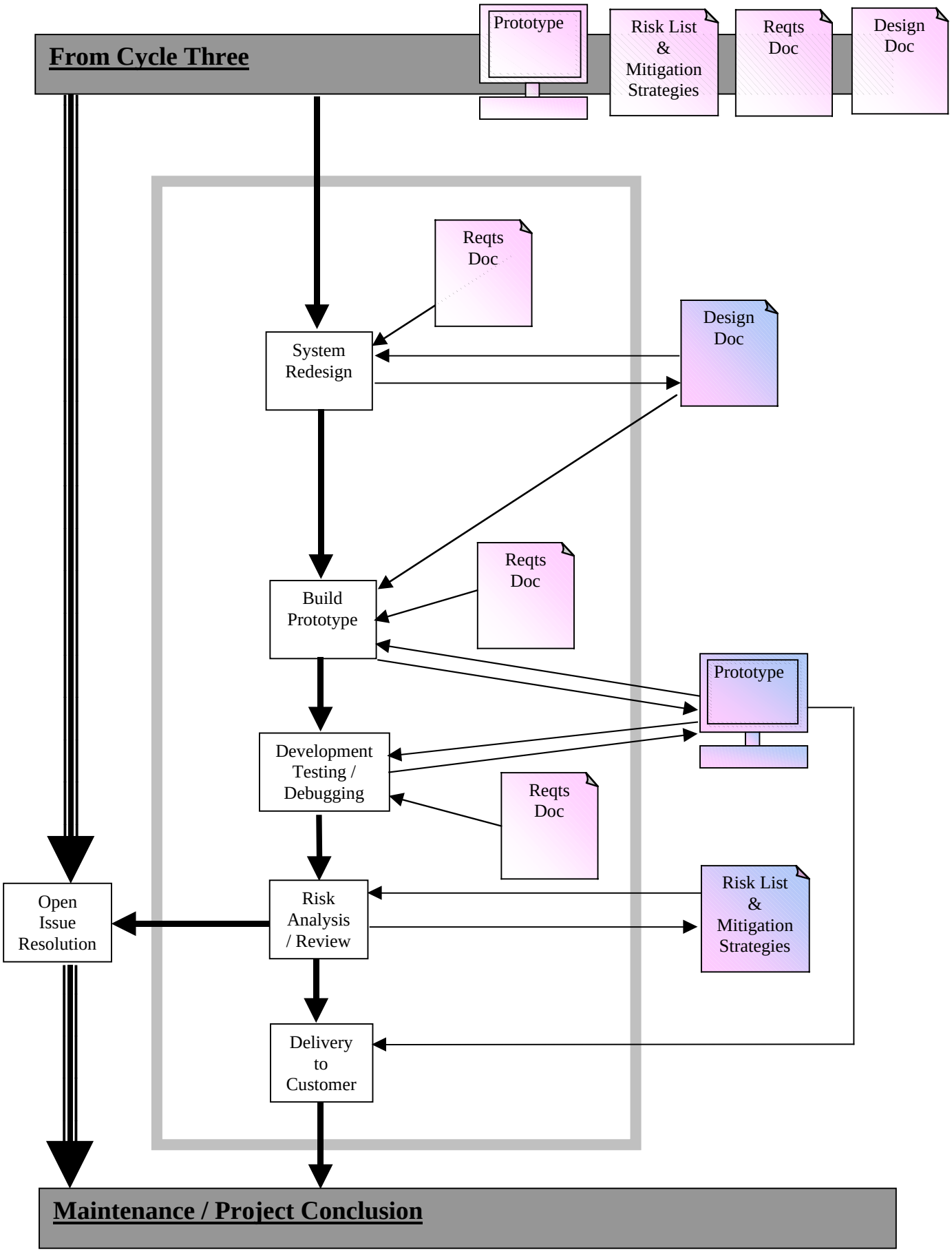
Note about the coloration given in the diagram:

Each cycle's work products are displayed in this diagram as having a color specific to that cycle. Work products produced in cycle 1 are faded to tan; work products in cycle 2 are faded to green, work products in cycle 3 are faded to pink, and work products in cycle 4 are faded to blue. This allows easier tracing and reference to a given work product. For example, one can distinguish between the requirements document created in cycle one, and the updated requirements document from cycle 2. A combination of the colors allows the reader to see that the work product was originally produced or modified in an earlier cycle and is updated in the current cycle. The documents appearing at the introduction bar of each cycle are work products that exist from earlier cycles. Finally, work products that are faded to gray (as shown in the first cycle) are those products that are continually modified throughout ALL cycles in the process.


Start of Project
SQA
Config. Mgmt.
Initial Customer Input
High level input (project description)
Update Business Plan
Business Plan
Create Business Plan
Update Project Plan
Project Plan
Create Project Plan
Maintain Security & Privacy Policies
Security & Privacy Policies
Create Security & Privacy Policies
Add Security & Privacy Reqts to Reqts Doc as needed
Gather Initial Reqts
Reqts Doc
Risk Mitigation
Risk Analysis / Review
Risk List & Mitigation Strategies
To Cycle Two


From Cycle One
Risk List & Mitigation Strategies
Reqts Doc
Prototype User Interface
User Interface Screens
Customer Evaluates UI
New & Modified Reqts
Updated Reqts Doc
Reqts Doc
Begin System Design
Partial Design Doc
Risk Mitigation
Risk Analysis / Review
Risk List & Mitigation Strategies
To Cycle Three


From Cycle Two
Risk List & Mitigation Strategies
Reqts Doc
Design Doc
Reqts Doc
Complete System Design
Design Doc
Reqts Doc
Build Prototype
Prototype
Development Testing / Debugging
Reqts Doc
Customer Evaluates Prototype
New & Modified Reqts
Updated Reqts Doc
Risk Mitigation
Risk Analysis / Review
Risk List & Mitigation Strategies
To Cycle Four


From Cycle Three
Prototype
Risk List & Mitigation Strategies
Reqts Doc
Design Doc
Reqts Doc
System Redesign
Design Doc
Reqts Doc
Build Prototype
Prototype
Development Testing / Debugging
Reqts Doc
Open Issue Resolution
Risk Analysis / Review
Risk List & Mitigation Strategies
Delivery to Customer
Maintenance / Project Conclusion

List of Document Templates

1. Statement of Work
2. Business Planning Document
3. Configuration Management Document
4. Project Plan
5. Software Quality Assurance Plan
6. Project Estimation Document
7. Privacy Policy
8. Security Policy
9. Requirements Specification
10. Design Specification
11. Review / Risk Meeting Agenda
12. Test Plan

[Project Name] Statement of Work

[Document Version Number]
[Date]

Document Author(s):
[Name]

Project Sponsor:
[Name]

Project Team:
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]

I. Introduction

This document justifies the project with a cost/benefit analysis and describes the business and project objectives.

Project Background
Describe any related projects that have contributed to the initiation of this project or those that should be worked in conjunction with this project. Note the status of these projects, their successes, challenges, and key contacts. Describe why and how this project was initiated and any related project history.

Related Documents
List documents associated with any related projects or the history of this project as described in *Project Background.*

II. Executive Summary

Requirements/Scope Summary
In paragraph form, provide an overview of the requirements and the scope.

Constraints/Issues
Describe any constraints or issues discovered thus far. This may include technical, schedule, resource, cost, or other issues that could not be resolved. Risks that are not strongly balanced with effective countermeasures may also be described here. Also identify any specific limitations within which the project must operate.

III. Project Overview

Business Objectives

Reference No.	Business Objective Description

Project Objectives
For each Business Objective, define one or more measurable Project Objectives.

Business Objective Ref. No.	Project Objective Ref. No.	Project Objective Description

IV. Justification

Benefit Analysis
What are the tangible, intangible and strategic benefits.

Cost/Benefit Analysis
Provide a quick cost/benefit summary.

Risk Analysis
What are the key project risk factors and the tasks necessary to manage or minimize these risks.

V. Document Revision History

Version	File version number.
Name(s)	Name of individual(s) responsible for the change.
Date	Date of change.
Change Description	Description of the changes made to the file.

[Project Name] Business Planning Document

[Document Version Number]
[Date]

Document Author(s):
[Name]

Project Sponsor:
[Name]

Project Team:
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]

I. Introduction

The purpose of this document is to provide guidance for the creation of a business plan for the eCommerce project. The business plan places the project in a market context and demonstrates the outlook, usefulness, as well as expected growth of demand for the system.

II. Executive Summary

This section should be created last. It needs to encompass the following four areas:

The Company, the Founders, the Vision

Tell who the company is and what it does to make money. Highlight the management – their accomplishments and aptitude. Sum up what management expects financial and operational output to be over the time covered by the business plan.

Market Opportunity

Why is this venture a good idea? What sort of market does the venture face in the future?

Use of Proceeds

How does the company plan to use proceeds? This section is often represented as a financial statement showing net proceeds, capital expenditures, and proceeds for working capital and reserves.

Financial Snapshot

Financial overview is provided in this section. It includes items such as revenues, cash flows, and selected performance metrics.

III. The Enterprise

This section is included to provide information about the product or service and its advantage in the market place. Here, you are 'selling' the project.

The Product

Business plan readers need to understand the product. Provide enough detail in order to communicate to the intended audience what you will produce.

Competitive Advantage

Put the product in the context of the market to demonstrate its advantages. A good way to show these advantages is in tabular form that compares the performance, cost, or other features to those of leading alternatives or substitutes.

IV. The Market

No matter how good of a product your group has, it needs a market to drive support for its creation. This section builds the case to the readers or investors of why they need to invest in the project. Here, you are 'selling' the market's potential.

The Industry

Give an overview of the industry of which your project will be a part. Include a background discussion of this industry, as well as industry trends that show future growth.

Market Size and Growth

Discuss your project's market. Note the size and growth rates that your team is facing.

Target Market

Quantify and describe the segment(s) for which you are gearing your project. Talk about segment prospects – their needs and wants and the trends that these segments are facing that may influence your project.

Competition

Talk about your competitors. Give key attributes of competitors or broad descriptions of competitor classes in this subsection.

Promotion and Advertising

What channels will you use to get people educated about your product? This subsection will provide a description of the advertising, education, and public relations that you will employ to drive demand. Describe the types and quantities of the media that will be used.

V. Operations

Use this section to talk about the items that are specific to the company's operations. In this part of the document, talk about any technological development efforts done to date, staffing, quality, or productivity if applicable. Give a description of the facilities and computer resources (information systems) that you may have. Take note, however, that management may not want some items disclosed to outside parties.

VI. The Organization

Include the names and titles of key officers in the company. Include brief, factual, relevant, career summaries for these officers. Officer compensation, supporting professionals (legal advisors, legal firms, and accountants), an organization chart, and employee headcount forecast may also be included if relevant.

VII. Business Risks

It is important to identify potential problems within the business context. It is often helpful to include some form of a risk mitigation strategy for these business risks.

VIII. Prospective Financials

Include financial information including assumptions made, balance sheets, income statements, and cash flow statements.

IX. Document Revision History

This section includes a list of significant changes that have been made to this document after 1.0 version has been submitted for assessment. The revision history should contain a dated list of revisions to the document consisting of: the date of each change, the person responsible for the change, and a description of the change. You should be able to trace changes to the individual who completed the modification. Changes are to be listed in reverse chronological order, recording the following information for changes:

Version	File version number.
Name(s)	Name of individual(s) responsible for the change.
Date	Date of change.
Change Description	Description of the changes made to the file.

[Project Name] Configuration Management Document

[Document Version Number]
[Date]

Document Author(s):
[Name]

Project Sponsor:
[Name]

Project Team:
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]

I. Introduction

The configuration management (see glossary) plan sets forth the guidelines to help control the changes inherent in software system development.

II. Managed Entities

- Requirements Document
- Privacy Policy
- Security Policy
- Project Plan
- Design Document
- Risk Analysis and Mitigation Documentation
- Prototype
- Any other documentation needed for historical or back-up purposes

III. Responsible Individual

The technical writer works to maintain and ensure configuration management practices. The project manager provides necessary support for ensuring that all team members assist with configuration management.

IV. Change Control and Version Management Policies

This section describes policies regarding the management of change and versioning within the software system. Areas to be addressed include:

- Where the files will be kept
- What process the group members must undergo to request that changes be made to the software
- How files will be maintained such that two individuals are not making changes to the same files without being aware of the other's changes
- Define standard naming conventions that may be applied so that versioning can be easily performed
- Who may access files and make changes

V. Document Revision History

Version	File version number.
Name(s)	Name of individual(s) responsible for the change.
Date	Date of change.
Change Description	Description of the changes made to the file.

[Project Name] Project Plan

[Document Version Number]
[Date]

Document Author(s):
[Name]

Project Sponsor:
[Name]

Project Team:
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]

I. Introduction

The Project Plan establishes team member roles and responsibilities. Establishes the schedule of project milestones and deliverables

II. Project Description

Provide a brief description of the goals and objectives of the project. Two paragraphs or one page is sufficient. Be sure to include a discussion of your clients. In other words, who have you made arrangements with to interview as part of your requirements gathering efforts?

III. Related Documents

List related information (and provide links) that will assist a reader in understanding any relevant background information that is useful for this project.

IV. Group Organization and Project Roles

Roles should be assigned to individual group members in order to avoid miscommunication and misunderstanding. Provide a brief description, according to roles, of each team member's specific responsibilities. The group must choose a project manager/group leader to organize and assess the progress of the project. The roles descriptions listed here are suggestions and should be modified as needed to fit your particular group.

Role	Individual(s)	Role Description	Type of Role
Project Manager (Technical)	[Name(s)]	1. Performs overall supervision of the project 2. Validates configuration management practices 3. Maintains project plan 4. Delegates requirements gathering 5. Performs implementation 6. Arbitrates problems as they occur 7. Member of the Software Engineering Group	Technical
Technical Writer	[Name(s)]	1. Oversees and maintains documentation 2. Takes meeting minutes and meeting documentation 3. Acts as the configuration management expert 4. Writes privacy policy and security policy and ensures that the work products adhere to these policies. 5. Performs implementation when necessary 6. Member of the Software Engineering Group	Management or Technical

Requirements Engineer	[Name(s)]	1. Leads requirements gathering efforts 2. Communicates with stakeholders, clients, end-users 3. Conducts interviews with stakeholders 4. Acts as interface to the customer 5. Documents requirements as well as policies elicited 6. Member of the Software Engineering Group	Management
Business Manager	[Name(s)]	1. Develops business plan, if appropriate, or revenue plan. 2. Maintains business plan 3. Member of the Software Engineering Group	Management
Chief Architect	[Name(s)]	1. Oversees design of overall system 2. Categorizes implementation work 3. Does resource planning 4. Performs implementation 5. Member of the SQA Group	Technical
Quality Assurance Expert	[Name(s)]	1. Designs test cases 2. Executes tests 3. Maintains fault / bug reports 4. Performs implementation 5. Member of the SQA Group	Technical
Lead Programmer	[Name(s)]	1. Delegates implementation work 2. Oversees defect resolution 3. Performs implementation 4. Member of the Software Development Group	Technical
Subject Specialist	[Name(s)]	1. Knows some special topic 2. Performs implementation 3. Member of the Software Development Group	Management or Technical
User Interface Design Specialist	[Name(s)]	1. Creates storyboard design (e.g. paper prototype or html editor generated design) 2. Member of the Software Development Group	Management

V. Schedule

Fill-in the assignment and completion dates for the provided (high-level) schedule for the project. The assignment date is the date on which a given task is planned to begin and the completion date is the date on which the task is scheduled to be finished. It may be helpful to supplement the table below with scheduling tools such as Gantt or Pert charts (see glossary) or a more-in depth schedule. When creating a schedule, remember that it is important to easily differentiate on what day a particular task should be complete. Tasks in the table are listed in the approximate order in which they need to be performed.

Task	Template Used	Assignment Date	Completion Date
Cycle 1			
• Obtain Initial Customer Input / Project Description			
• Create Business Plan	Business Plan		
• Create Project Plan	Project Plan		
• Write Privacy and Security Policies			
• Perform Initial Requirements Gathering	Requirements Document		
• Hold Risk Analysis and Review Meeting	Review and Risk		
Cycle 2:			
• Perform User Interface Prototyping			
• Start Initial System Design	Design Document		
• Have Customer Evaluate User Interface			

• Modify Requirements	Requirements Document		
• Hold Risk Analysis and Review Meeting	Review and Risk		
Cycle 3:			
• Complete System Design	Design Document		
• First Prototype Development, Debugging, and Testing			
• Have Customer Evaluate First Prototype			
• Modify Requirements	Requirements Document		
• Hold Risk Analysis and Review Meeting	Review and Risk		
Cycle 4:			
• Final Prototype System Redesign	Design Doc		
• Final Prototype Development and Debugging			
• Extensive Development Testing			
• Hold Risk Analysis and Review Meeting	Review and Risk		
Throughout the Cycles:			
• Configuration Management	Configuration Mgmt		
• Privacy and Security (Policy) Management			
• Quality Assurance	SQA		
• Project Planning Updates	Project Plan		

VI. Document Revision History

Version	File version number.
Name(s)	Name of individual(s) responsible for the change.
Date	Date of change.
Change Description	Description of the changes made to the file.

[Project Name] Software Quality Assurance Plan

[Document Version Number]
[Date]

Document Author(s):
[Name]

Project Sponsor:
[Name]

Project Team:
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]

I. Introduction

The document describes a project team's SQA goals and responsibilities. It also includes the description of where the SQA group is placed within a project (including the reporting structure and the manner in which SQA group will interact with software engineering teams).

II. Quality Attributes

The software quality plan clearly states what quality attributes (e.g. safety, security, reliability, resilience, robustness, understandability, testability, adaptability, modularity, complexity, portability, usability, reusability, efficiency, learnability, maintainability) are more important to your project. You should justify which attributes you choose to emphasize. This provides a basic ideal that all practitioners can target throughout development. Be aware that you may have to give up higher levels of one attribute (such as efficiency) for higher levels of another (such as usability).

III. SQA Tasks

Provide a brief description of all SQA tasks and assign responsibility for each if possible. Software Quality Assurance tasks are those tasks that establish team procedures and standards to increase quality in the software. A very high-level summary of SQA is:

→ step 1: develop the product
→ step 2: assess product (software) quality
→ step 3: if the quality is not sufficient, improve the process and return to step 1, else standardize the process.

Tasks may include, but are not limited to:

- definition of when and how software inspections will occur to find errors in the design or code
- development of product standards
- development of process standards
- development of documentation standards
- monitoring the development process to ensure that both the process and product standards are being followed
- monitoring progress of the project to provide management information
- defining when and how quality reviews will occur to find differences between the specification and design, code, or documentation
- collection of metrics if necessary and desirable
- identification of procedures for error reporting and tracking

IV. Document Revision History

Version	File version number.
Name(s)	Name of individual(s) responsible for the change.
Date	Date of change.
Change Description	Description of the changes made to the file.

[Project Name] Project Estimation Document

[Document Version Number]
[Date]

Document Author(s):
[Name]

Project Sponsor:
[Name]

Project Team:
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]

I. Introduction

The estimation template provides methods for project estimation including software work products , software effort, and critical computer resources.

II. Size and Effort Estimate

By estimating the size of the product you plan to build, you can better judge the amount of work required to build it. We recommend using a method taken from extreme programming (see glossary) literature where you divide the work to be done into tasks that take approximately one full workday to complete. The project plan that you have established will assist you in determining what needs to be done next and the dates whereby things will need to be completed. Thus, size estimation is planned on small tasks rather than the larger task blocks. Divide the major process steps into small tasks and allow each group member 'own' the tasks for which he or she is responsible.

III. Critical Computer Resources Estimate

In this section, estimate the hardware and the software that you will need to finish the eCommerce application development. This can be done by listing the estimated resources in a table such as the following:

Category	Type	Example
Hardware	Platforms	Workstation or PC & the number of such machines that you will need
	Internet connections	*Example*: DSL, modem, school network, etc.
Software	Operating Systems	*Example*: Windows family, Linux, Unix, Mac, etc.
	Development Tools	
	Databases	

IV. Document Revision History:

Version	File version number.
Name(s)	Name of individual(s) responsible for the change.
Date	Date of change.
Change Description	Description of the changes made to the file.

[Project Name] Privacy Policy

[Document Version Number]
[Date]

Document Author(s):
[Name]

Project Sponsor:
[Name]

Project Team:
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]

I. Introduction

This document is intended to help create the parts of a privacy policy that will be required as a result of the new project deliverable. Documents new and/or modified privacy policies which must be adhered to by the system. Describes the kinds of information collected by the web site and the way that information is handled, stored, and used.

II. Privacy Policy

A privacy policy should, at a minimum, address the Fair Information Practice Principles.

A. Notice/Awareness

A privacy policy must be easy to find and understand. The policy must be available prior to collecting or requesting individually identifiable information. State clearly what is being collected and how it will be used.

B. Choice/Consent

Consumers should be given the opportunity to choose how individually identifiable information collected from them online may be used.

C. Integrity/Security

State how the company is taking appropriate measures to assure data reliability and protect it from loss, misuse or alteration.

D. Access/Participation

Consumers should be able to access the information being collected about them. State how inaccuracies in individually identifiable information, such as account or contact information, may be corrected.

III. Related Documents

List related information (and provide links) that will assist a reader in understanding any relevant background information that is useful for this project.

IV. Document Revision History

Version	File version number.
Name(s)	Name of individual(s) responsible for the change.
Date	Date of change.
Change Description	Description of the changes made to the file.

[Project Name] Security Policy

[Document Version Number]
[Date]

Document Author(s):
[Name]

Project Sponsor:
[Name]

Project Team:
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]

I. Introduction

This document is intended to help create the parts of a security policy that will be required as a result of the new project deliverable. According to the new system requirements, it will establish an associated plan of how both internal and external users interact with the new system, how the computer architecture topology will be implemented, and where computer assets will be located**.** Addresses security goals, risks, levels of authority, procedures for addressing security breaches, and other details impacting system security.

II. Security Policies

A security policy consists of many specific policies. The following are some examples that might be required for the project (note that this is just an example list):

- Identification and Authentication Policy
- Encryption Policy
- Awareness and Education Policy
- Password Policy
- Remote Access Policy
- Database Access Policy
- Appropriate Use Policy

Each policy will be structured similar to the following:

Statement of Purpose
Provide a brief statement that explains why the policy is necessary.

Scope
Provide a description of the policy's applicability.

Glossary
List any definitions or explanations that will assist the reader in fully understanding the policy.

Policy
The actual policy statement that explains the rules the policy will implement and various roles and responsibilities.

III. Related Documents

List related information (and provide links) that will assist a reader in understanding any relevant background information that is useful for this project.

IV. Document Revision History

Version	File version number.
Name(s)	Name of individual(s) responsible for the change.
Date	Date of change.
Change Description	Description of the changes made to the file.

[Project Name] Requirements Specification

[Document Version Number]
[Date]

Document Author(s):
[Name]

Project Sponsor:
[Name]

Project Team:
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]

I. Introduction

The requirements document specifies the services that the system will provide and the constraints under which the system must operate.

II. Functional Requirements

Enumerate all functional requirements (see glossary) in this section. It is generally a good idea to organize the functional requirements according to the modules into which the system has been decomposed by the system architect. Don't forget to provide traceability information such as where the requirement originated and a unique identifier, what the priority is, and a definition of different priority levels.

Example:

V.1. Communication with Server.

V.1.1.

The system shall be able to communicate with the Zephyr server.

Description: Messages, location of other users, and class subscriptions must all be handled through communication with the server.

Origin: Use cases III.2.1., III.2.2., III.2.5., III.2.12., III.2.13., III.2.14., and III.2.15. Customer interview from November 11, 2000.

Priority: 2

III. Nonfunctional Requirements

Enumerate all nonfunctional requirements (see glossary) in this section. Note that in eCommerce systems, privacy and security requirements are extremely important. The next section of this document will contain all of these requirements, but it is worth mentioning them here as nonfunctional requirements. Again, it is best to organize the nonfunctional requirements according to the modules into which the system has been decomposed by the system architect. Don't forget to provide traceability information such as where the requirement originated and a unique identifier, what the priority is, and a definition of different priority levels.

Example:

VI.1. Timing

VI.1.1.

WindowGrams sent by WinZephyr shall be received at the destination in an amount of time comparable to Unix Zephyr.

Description: The time to receive a WindowGram is described as nearly instantaneous. Messages consisting of 200 characters (roughly three lines of text) shall be sent to and received from the server in an average of two seconds.

Origin: Zephyr on Athena Manual.

Priority: 3

IV. Security and Privacy Requirements

Due to the sensitive nature of many software systems, security and privacy requirements (see glossary) must be highlighted and enforced. Enumerate security and privacy requirements in this section. Don't forget to provide traceability information such as where the requirements originated and a unique identifier, what the priority is, and a definition of different priority levels. Phrase as you do nonfunctional requirements.

V. Requirements Traceability Matrix

Provide a cross-reference matrix showing related requirements as shown in the example below. Please note that some software engineering tools will generate a traceability matrix such as this for you; thus, this section may not be applicable for your project.

		Is dependent on requirement			
Requirement		Req1	Req2	Req3	Req4
	Req1		**X**		
	Req2				
	Req3				**X**
	Req4	**X**			

VI. Development and Target Platforms

Describe in full detail the expected development and target platforms including software/hardware types, versions, etc...

VII. Document Revision History

Version	File version number.
Name(s)	Name of individual(s) responsible for the change.
Date	Date of change.
Change Description	Description of the changes made to the file.

[Project Name] Design Specification

[Document Version Number]
[Date]

Document Author(s):
[Name]

Project Sponsor:
[Name]

Project Team:
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]

I. Introduction

This document sets forth the description of the software solution that meets user's requirements within the constraints of the NCSU eCommerce Practicum environment. The design document recounts 'how' the software will work and details the software architecture as well as the conceptual and technical design for the system. Includes description of components as well as data structure and data flows.

II. System Overview

Provide an overview of the system in this section, detailing the subsystems and modules of your system design. Add a brief subsystem and module overview to this section. Include a diagram showing the system structure.

III. Design Description

This section presents the architectural design (see glossary) of your system.

Data Flow/Structure

Describe the important data flow paths of your design and provide a diagram of the flow of data.

Software Structure

A diagram should be included in this section to show the architectural design of your system. The diagram must show the relationship among the subsystems and modules of your system and should provide a hierarchy for control and processing. Remember that "architectural design" includes discussion of data and control flow.

IV. Subsystem and Module Specification

Specify the design for each subsystem and module in your software system. All functions should be described in detail. (Detail means providing a brief description of the subsystem and more detailed information about the modules including a processing narrative, a description of the interface(s) associated with this module, references to other modules used by this module, any data structure internal to the module, and any additional information (comments, restrictions, or limitations) about the module. Feel free to use diagrams in this section to help describe the subsystems/modules.

V. File Structure and Global Data

Describe any information pertaining to database requirements and the types of data that will reside in the database. (Identify data, describe organization of data, specify types of data, note any constraints or valid/invalid ranges of data and any other information you might deem important. This data might be data maintained in a file, database, or in common memory.)

VI. Interface Design

Human-Machine Interface Specification

For example: GUI (graphical user interface) designs

Human-Machine Interface Design Rules

Rules that guide you in the creation of your Interface Design to maintain consistency.

External Interface Design

Interfaces to External Data

Interfaces to External Systems or Devices

Internal Interface Design Rules

Rules that specify how modules, functions, methods, etc. interface with other modules, functions, or methods.

VII. Requirements Cross Reference

List any additional/changed requirements that are identified during the design process. Also provide a cross reference of design to the requirements in the requirements document to show that all the requirements are being met by your design. Discuss any requirements that are not fulfilled by the design.

VIII. Document Revision History:

Version	File version number.
Name(s)	Name of individual(s) responsible for the change.
Date	Date of change.
Change Description	Description of the changes made to the file.

[Project Name] Review/Risk Meeting Agenda Document

[Document Version Number]
[Date]

Document Author(s):
[Name]

Project Sponsor:
[Name]

Project Team:
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]

I. Introduction

This document provides ordered list of topics that must be discussed during the review/risk meeting held at the beginning of each evolutionary cycle. Each item addressed during the meeting and the resulting mitigation strategies are documented in the resulting meeting minutes.

II. Meeting Topics

Review

A brief description of the tasks that each individual completed during the previous cycle is given, along with a description of what tasks he or she faces at the present and in the future cycle. Problems that the individuals encountered during the past cycle are also reviewed. The purpose of this review is to keep all group members motivated, make sure that all are aware that their efforts will not go unnoticed by the group, and to give the entire team a sense of what each person is doing and how the project is progressing. Also, during the review, the group measures its progress of how well it has handled the risks placed under a mitigation strategy from the last risk meeting. While documentation of the reviews is not necessary, the project plan must reflect the status of the project as it exists as provided in the reviews.

Compare newly generated requirements with already established requirements and policies

As a group, hold an inspection of the newly generated requirements (especially those generated by the customer). Discuss the feasibility of the requirements. Do they conflict with existing requirements, existing policies (security, privacy, governmental), or other newly-generated requirements? If so, mark this as a risk that must be negotiated and mitigated.

Define the risks and problems associated with the project

Derive an initial risk (see glossary) list from a brainstorming session, knowing that risks may come from the system developers, users, application domain, problem domain, computer system, and/or the development environment.

Update the risk list at the beginning of each prototyping cycle with new risks that the group may be facing. Pay close attention to risks that are created when the customer significantly adds or modifies requirements in such a way that the functionality of the system will drastically change from what was previously agreed. (This is known as ‘functionality creep’). This means that the group should review all added and modified requirements to determine if functionality creep is a risk that the team is facing.

Evaluate these risks to derive their consequences

This process is subjective. Some areas of consequence may fall from deteriorated social relations, process problems, product defects, and/or exceeded resource limits.

Assign priorities to the risks to find which risks must be emphasized and mitigated.

Negotiate the severity and the probability of each risk on a scale from a low 1 to a high 5. The severity and probability values should be multiplied together to form each risk's score. The higher the product, the more high-risk the issue.

Complete resolution strategies for risks of which there is the most concern.

Choose the (top four or five) higher-risk issues and develop mitigation strategies for those issues. Strategies may fall from the areas of improving social, organizational, and technical setting of the project; developing a pilot prototype; providing different options; or limiting prototype size, among others. Documentation of these strategies must be maintained.

In particular, it may be necessary to negotiate requirements changes with the customer to mitigate requirements creep risks as mentioned above. You might relax the schedule, increase your allowed cost, reduce the system quality, add resources, or reduce the scope by ignoring those requirements that do not make sense from a development standpoint. This is a negotiation process that must take place between the developers and the customer if you face the risk of functionality creep.

[Project Name] Test Plan

[Document Version Number]
[Date]

Document Author(s):
[Name]

Project Sponsor:
[Name]

Project Team:
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]
[Name] [Role]

I. Introduction

This document details the testing process, items to be tested, testing schedule and test recording procedures. This serves as the plan for testing all software artifacts as well as the reporting of test results.

II. Test Items

Describe items and features to be tested and those not requiring testing (and why).

III. Approach for Testing

Specify the major activities, techniques and tools which will be employed to test each feature.

- Provide sufficient detail to facilitate estimation of project scheduling, costs, etc.
- Specify techniques that will be used to assess how comprehensive testing and enumerate test completion criteria.
- Specify techniques used to trace requirements.

IV. Pass / Fail Criteria

Specify the pass / fail criteria for each test.

V. Testing Deliverables

Specify the planned testing deliverables which may include:

- Test Design Specification
- Test Case Specification
- Test Procedure Specification
- Test Log
- Test Incident Report
- Test Summary Report
- Test Input and Output Data

VI. Environmental Requirements

Specify the environmental needs for conducting tests:

- Hardware, communications and system software, other supplies, etc.
- Level of security
- Testing tools

VII. Staffing

Specify testing responsibilities, staffing and training needs.

VIII. Schedule

Specify testing schedule.

IX. Risks and Contingencies

Specify any potential risks and plans for mitigating, addressing and/or resolving those risks.

X. Approvals

List any approvals / signatures required to sign off on test results.

XI. Document Revision History:

Version	File version number.
Name(s)	Name of individual(s) responsible for the change.
Date	Date of change.
Change Description	Description of the changes made to the file.

Glossary

The following definitions were taken from Software Engineering, 5th edition, 1995, by Ian Sommerville unless otherwise denoted by *. Definitions marked with * contain sources included with the definitions.

architectural design – The initial design process of identifying an application's sub-systems and establishing a framework for sub-system control and communication.

configuration management – The process which controls the changes made to a system and manages the different versions of the evolving software product.

quality assurance / SQA – The establishment of organizational procedures and standards which lead to high-quality software.

nonfunctional requirement – A constraint placed on the system or on the development process.

functional requirement – A requirement that describes a system service or function.

***privacy requirement** – Requirement that focuses on the privacy aspects of the application.

***security requirement** – Requirement that focuses on the security aspects of the application.

***evolutionary prototyping** – a process in which a prototype ultimately becomes the operational system and where some parts of the system may be re-designed and re-coded using a more formal approach. (from http://www.doc.mmu.ac.uk/STAFF/E.Ferneley/SAD/T23/proto.htm)

Waterfall model – A software development model containing the phases of requirements definition, system and software design, implementation and unit testing, integration and system testing, and operation and maintenance in which the process 'cascades' from one phase to another.

***Capability Maturity Model** (CMM) -- A five-level framework for how an organization matures its software process from chaotic processes to mature, discipline software process. (from http://sern.ucalgary.ca/~wangyu/SENG/621/presentation/wuw/tsld020.htm)

***eCommerce** -- Conducting business online

spiral model – A software development model that has its basis in explicitly recognized risk. The model takes the form of a spiral in which each loop in the spiral represents a management-defined phase of the software process; and each loop in the spiral is split into the four sectors of objective setting, risk assessment and reduction, development and validation, and planning.

throw-away prototyping – A development activity in which the objective of the evolutionary development process is to understand the customer's requirements by concentrating on experimenting with the parts of the customer requirements which are poorly understood.

project plan – A plan, drawn up at the start of a project, which should be used as the driver for the project. This plan is not static but must be modified as the project progresses and better information becomes available.

***business plan** – A plan which helps to place the project in a market context and demonstrate the outlook, usefulness, and growth of demand for the project.

***(chief) architect** -- Lead technical person on team; understands documents and overall organization of system; leads programming effort; sees the big picture.

risk – Informally, something which can go wrong. Risk is a consequence of inadequate information; therefore risks are resolved by initiating some actions which discover information that reduces uncertainty.

***extreme programming** – (XP) is a pragmatic approach to program development that emphasizes business results first and takes an incremental, get-something-started approach to building the product, using continual testing and revision. (from http://iroi.seu.edu.cn/books/whatis/xp.htm)

***Gantt chart** -- A horizontal bar chart that graphically displays the time relationships between the different tasks in a project. (from http://www.eng.uci.edu/~ghubbard/mae189/gantt.html)

PERT chart – A sophisticated form of activity chart in which pessimistic, likely, and optimistic estimates are made for each task in a software project.