PROJECT WORK PLAN

Organization: CERN – LCG Project

SEAL
Common Core Libraries and Services for LHC Applications

Document Revision #: 0.1

Date of Issue: 10.01.2003

Project Manager: Pere Mato

Approval Signatures

Approved by: Business Project Leader	Approved by: IM/IT Project Leader
Prepared by: Business Project Manager	Prepared by: IM/IT Project Manager
	Reviewed by: Quality Assurance Manager

Table of Contents

Document Change Control

This section provides control for the development and distribution of revisions to the Project Work Plan up to the point of approval.

Revision Number	Date of Issue	Author(s)	Brief Description of Change
0.1	7.01.2003	P. Mato	Initial Draft

1. Project Overview

The common core libraries and services (SEAL) project is initiated from the recommendation of the Blueprint RTAG [1] to provide the basic software infrastructure for other LCG application area projects and in common for the LHC experiments.

1.1 Purpose, Scope, and Objectives

The purpose of the project is to provide the software infrastructure, basic frameworks, libraries and tools that is common among the LHC experiments. The project should address the selection, integration, development and support of foundation and utility class libraries. These utilities cover a broad range of unrelated functionalities and it is essentially impossible to find a unique optimum provider for all of them. They should be developed or adapted as the need arises. In addition to these foundation and utility libraries, the project should develop a coherent set of basic framework services to facilitate the integration of LCG and non-LCG software to build coherent applications.

The scope of the SEAL project is basically the scope of the LCG Applications Area. The Applications Area expected scope includes common applications software infrastructure, frameworks, libraries, and tools; common applications such as simulation and analysis toolkits; and assisting and supporting the integration and adaptation of physics applications software in the Grid environment

The SEAL project should provide a **coherent** and as complete as possible set of core classes and services in conformance with overall architectural vision described in the Blueprint RTAG.

The two main deliverables of the project are: *Foundation Class Libraries* and *Basic Framework services*.

Foundation Class Libraries

- Basic Types. These libraries are low level fairly independent class libraries to complement the standard basic types (int, float …). In addition to the basic types which are applicable to a wide range of applications there will be basic types specific to HEP (e.g. LorentzVector).
- Utility libraries. These utilities cover a broad range of unrelated functionalities which makes sense to re-use across LCG projects. These will be developed or adapted as the need arises.
- System libraries. These are the libraries that should provide an operating system isolation layer. Instead of each LCG project or experiment developing its own interface to the system services a common one should be provided.

- Domain specific foundation libraries. Many other core libraries will be needed to implemented high level services and software components in many domains (simulation, interactive analysis, etc.). The project should provide support for such libraries.

Basic Framework Services

A coherent, integrated set of core infrastructure and core services supporting the development of higher level framework components and specializations should be addressed by the project. An initial set of basic services have been identified in the Blueprint RTAG.

- Component model. Develop or adapt a basic set of mechanisms and base classes for managing creation of objects (factories), lifetime, multiplicity and scope, component communication and interface discovery.
- Reflection. Reflection is the ability to query a class about its internal structure at run time. This finds applications in object streamers, object browsers, rapid prototyping, etc. Since C++ does not provide this kind of functionality, the project should develop an object dictionary to provide reflection functionality by complementing the native C++ language features.
- Plugging management. In the LCG architecture a plug-in is a logical module that encapsulates the concrete implementation of a given service. The plugging manager is responsible for loading, activating, deactivating and unloading plug-ins at run-time.
- Other framework services. Develop basic framework services for message reporting, exception handling, component configuration, "event" management, object "white board", etc. Other services candidate to be developed will be identified by the other LCG applications area projects.
- Scripting. Provide the basic infrastructure to support scripting. In particular, bindings for Python and CINT of the basic services will be needed to provide a "component bus" that allow easy integration of components providing a wide variety of functionality, possibly implemented in a variety of languages.
- Distributed computing, Grid. Provide a common interface to the Grid services to be used by the other LCG application area projects.

The SEAL project provides core software services and support for foundations libraries to the other LCG application area software projects such as object persistency project (POOL), math libraries project, physicists interface project (PI). It is expected that the LHC experiment frameworks, which cover additional areas not in the scope of the LCG application area such as event processing and algorithm scheduling framework, will also benefit of the SEAL deliverables by facilitating their integration with the LCG provided functionality. Other HEP software projects not specifically for LHC, with perhaps a wider domain of applicability, can also take advantage of the basic framework services that will be provided by SEAL.

It is expected that new needs and requirements will be uncovered later during the execution of the project. It is also expected that software components or utility libraries originally developed for a specific project will be migrated to the SEAL project as soon as they became candidates for more general re-use.

1.2 Assumptions, Constraints and Risks

The SEAL project is based on the following assumptions:

- Most of the core software which is going to be delivered in a coherent packaging by the project exists in one form or another in the experiment's core software. It is assumed that most of the work will be in re-packaging existing pieces of software, since the resources available are not sufficient for a complete development of whole the core software.
- The LHC experiments are committed in using the LCG software when available. It is essential for the success of the project that they provide fast feedback of the deliverables.

Here are the known constrains and risks of the project:

- The project will deliver the components as soon they are available. The priorities will be set by the users (other LCG projects and LHC experiments).
- The quality of the core software delivered by the project must be superior to any other software component of application based on it.
- We will re-use as much as possible existing software. This implies to make compromises and adaptations to achieve the required level of coherency and conformance to the architectural vision already established.
- Any project adopting the software developed by this project will need to be adapted and will required to replace their own software elements with the ones functionally equivalent provided by the project. This will certainly imply some period of instability for the experiment applications.

1.3 Project Deliverables and Schedule

As already mentioned before, the project will deliver the software incrementally. The first releases will fulfill the needs of the other LCG projects. The schedule of the delivery of the first version is summarized as follows:

- Release *V1 alpha* defined as essential functionality sufficient for the other existing LCG projects by end March 2003. Frequent internal releases before this date.
- Release *V1 beta* defined as essential functionality sufficient to be adopted by experiments frameworks by end June 2003

The main milestones of the project are:

- 2002/11/30 Define the V1 SEAL software

- 2002/12/1 Prototype object dictionary service
- 2003/1/15 Establish external software decision process. Establish the process and policies by which decisions are made on what external software is to be used by the LCG applications area.
- 2003/1/31 Complete the initial SEAL work plan. Complete the initial SEAL work plan for submission to the SC2. Should cover (at least) the content and implementation plan for SEAL V1.
- 2003/3/31 SEAL V1 essentials in alpha. The most essential elements of the V1 SEAL suite (as requested by projects needing to use them) are available in alpha.
- 2003/5/31 Grid enabled services defined. The SEAL services which must be grid-enabled are defined and their implementation prioritized.

1.4 Budget Summary

The computing infrastructure required to run the project, not including the developer's desktops, such as test servers, build servers, code repositories, web site are assumed to be provided by the SPI project together with the CERN IT and EP divisions. The staff joining the SEAL project will keep their desktop hardware, which is generally provided by their institute.

1.5 Evolution of the Plan

The structure of this Project Plan is in compliance with the recommendations of IEEE Std 1058-1998.

- The project plan should be revised every six months in order to reflect the changes in requirements and priorities, and the evolution in the level of staffing. A good moment to revise this plan would be just after the release of the of v1 beta during summer 2003.
- As soon as the SC2 committee agrees to this planning, it will be used as a baseline for project tracking. The plan will be controlled and tracked quarterly.
- The updates will be presented to the SC2 together with the tracking of the existing plan.

1.6 References

[1] Report of the LHC Computing Grid Project Architecture Blueprint RTAG, CERN-LCG-2002-022, (http://cern.ch/lcg/SC2/RTAG8/finalreport.doc)

1.7 Definitions and Acronyms

LCG

LHC Computing Grip project (http://cern.ch/lcg)

SC2

Software and Computing Steering Committee. Body constituted of representations from LHC experiments to provide requirements and monitoring to LCG sub-projects.

RTAG

Requirements and Technology Assessment Group. Launching an RTAG on a given domain is the mechanism for defining potential common projects within the LCG project.

SPI

LCG Software Process and Infrastructure project.

2. Project Organization

2.1 External Interfaces

? There are several projects already existing in the LCG Applications Area and SEAL is one of them. Figure 1 shows these projects and their interfaces.

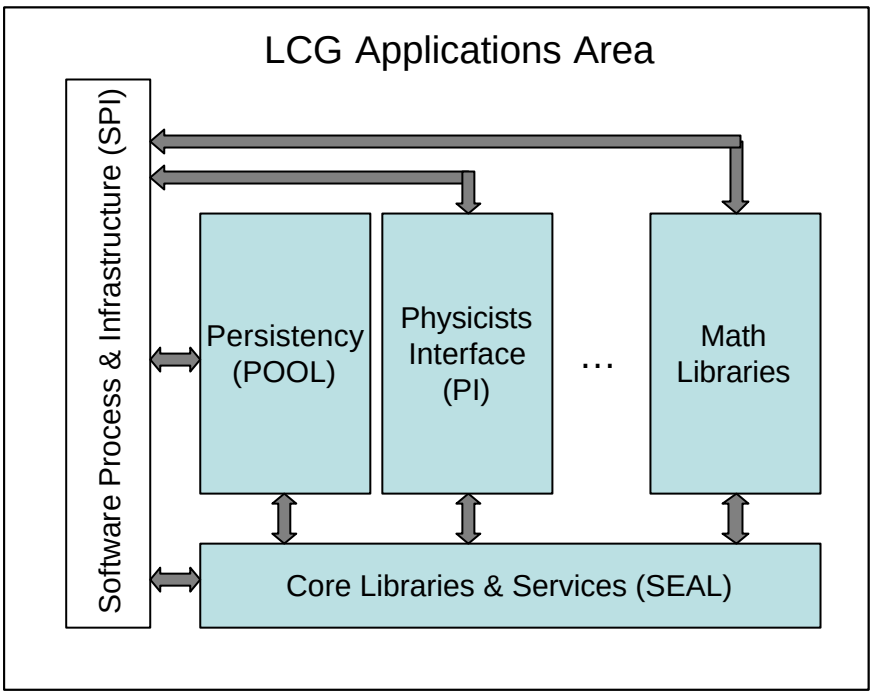

Figure 1 Organisational chart depicting the projects of the LCG Applications Area and their interfaces (arrows)

? Project management. The project reports to the LCG Applications Area manager (T. Wenaus), to the LCG project leader (L. Robertson) and to the SC2 committee (chaired by M. Kasemann).

? Line management. The CERN IT and EP staff of the project reports to their corresponding CERN group leader.

? Customers. In its current definition, the users of the deliverables of the project are all the LCG software projects and any other CERN or non CERN experiment interested in using part or the entire set of core libraries and services.

2.2 Internal Structure

Currently we do not see a need for a complex internal structure of the project. The project will be organized as a number of *work packages* (WP) producing a one or more deliverables. Each WP will have a responsible reporting to the SEAL project leader.

During the definition phase of the work packages with their deliverables and responsibilities, the project team will work as a single team without any further structure.

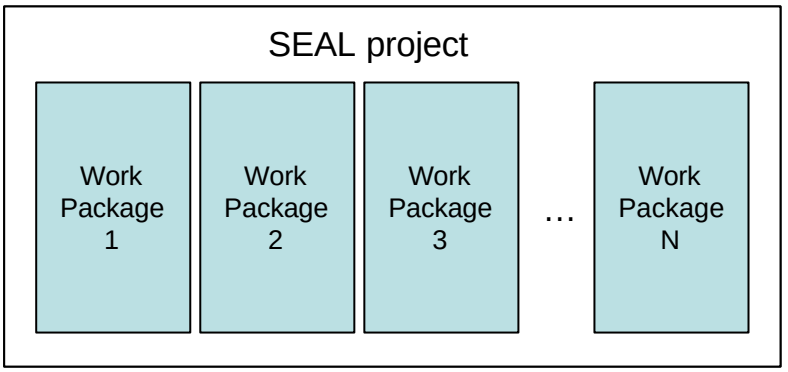

Figure 2 Internal structure of the SEAL project

2.3 Roles and Responsibilities

The roles currently defined in the project are:

- **Project Leader**. The project leader role allocates resources, shapes priorities, coordinates interactions with customers and users, and generally keeps the project team focused on the right goal.
- **Work Package Manager**. Responsible for delivering the agreed products of the work package.
- **Developer**. The developer is involved in designing and developing software in all process phases. We do not distinguish from designers, implementers, testers, etc.

3. Managerial Process Plans

This section of the Project Plan specifies the project management processes for the project. This section defines the plans for project start-up, risk management, project work, project tracking and project close-out.

3.1 Start-up Plan

3.1.1 Estimates

This plan has been produced by the project leader in collaboration with the initial project team based on the perception of the objectives and the existing experience of similar projects of these characteristics developed in the various core software projects in the LHC experiments.

Weekly status and progress meetings will provide data for improvement of these estimates and for planning the second release.

The schedule will be re-estimated after 6 months or in the event of major changes in the policies regarding the project objectives, such as urgent needs of new LCG projects currently at the RTAG definition phase.

3.1.2 Staffing

? The initial idea concerning staffing is to start with rather small team to define the components that need to be developed and establish the development process and style. The software skills of this initial team should cover various computing domains. In addition, the team members should have strong links to the experiments (our final clients) to ensure that the project deliverables will integrate well with the experiment application software and therefore be easily adopted by them. After this initial phase, the project team will be expanded incorporating new developers and work package managers to develop the final high quality products. The rational behind this approach is that is it should be easy to agree on a development style and convey the architectural vision to a small number of people rather than a big team geographically distributed.

? It is desirable that the staff working in the project have a several years of experience in this kind of core software. Ideally, it would be nice if the same people that did this kind of work already in the experiment's core software share their experience and build the SEAL software in common.

? The current level of staffing in this initial phase of the project is about 3 FTE. This should ramp to about 8 FTE during the next 6 months just about the time of the release of version v1.

? The expected level of commitment of the staff during the first phase should be more than 50%. This is a requirement have soon some progress and in that way to establish quickly the team dynamics.

? The sources of staff

- Participation of the LHC experiments projects to the LCG project from CERN and other Institutes.
- LCG funded posts.

? The following table shows the list of current staff of the project and their current commitment to SEAL.

Table 1 Roles and Staffing

Role	People	Commitment	Comments
Project Leader	Pere Mato	50%	also in LHCb
Developer	Jacek Generowicz	70%	also in Geant4
Developer	Massimo Marino	70%	also in ATLAS
Developer	Lorenzo Moneta	50%	also in LCG/SPI
Developer	Stefan Roiser	50%	also in LHCb
Developer	Lassi Tuura	50%	Also in CMS

3.1.3 Resource Acquisition

The resources to be acquired to fulfill the estimated of 8 FTE will happen in the next 6 months. The acquisition mechanism is mainly based on the goodwill of the people (mainly from the experiment's projects) to join the project if they are interested in developing some of the components.

3.1.4 Project Staff Training

No specific training is required. It is assumed that the interested people joining the project will have sufficient experience in the domain that training should not be necessary.

3.2 Work Plan

3.2.1 Work Breakdown Structure

The initial work packages of the project have been defined and are shown in Table 2 together with a short summary of each one. Later in this section, each work package is developed with a bit more detail and the deliverables are listed indicating if it is foreseen for the first release of the software or later releases.

Table 2 SEAL Work Packages

WBS	Name	Summary
1	Foundation and Utility libraries	Set of foundation and utility libraries including an operating system abstraction layer to be used by all other LCG software projects.
2	Component Model and Plug-in Manager	Develop or adapt a basic set of mechanisms and base classes for managing creation of objects (factories), lifetime, multiplicity and scope,

		component communication and interface discovery. Develop the plugging manager is responsible for loading, activating, deactivating and unloading plug-ins at run-time.
3	LCG Object Dictionary	Develop an object dictionary to provide reflection functionality by complementing the native C++ language features. The two aspects: population of the dictionary and reading the dictionary information through the reflection interface are considered.
4	Basic Framework Services	Develop basic framework services for message reporting, exception handling, component configuration, "event" management, object "white board", etc. Other services candidate to be developed will be identified by the other LCG applications area projects.
5	Scripting Services	Provide the basic infrastructure to support scripting.
6	Grid Services	Provide a common interface to the Grid services to be used by the other LCG application area projects
7	Education and Documentation	Provide a coherent documentation for components either developed in the project or adopted from third parties. Prepare tutorials and provide help for integrating project deliverables into experiments applications.

1 Foundation and Utility libraries

Provide class libraries to complement the standard types and utility libraries to be used by all the LCG software projects.

? Main activities and tasks

- Inventory of existing utility classes
- Provide support for *Boost* library. Boost is a utility library (open source project) that is a strong candidate to standardize on. It is intended to become part of Standard Library (STL).
- Participation to CLHEP project as recommend in the Blueprint RTAG. Prepare proposal for its evolution.
- Develop SEAL *utility* and *system library* complementary to *Boost* and STL using from existing code from various libraries in use in the experiments (ClassLib, Gaudi, HepUtilities, etc.).
- Establish guidelines for selecting external libraries

? Proposed v1 deliverables

- SEAL utility candidates inventory
- Support *Boost* library (installation, documentation, etc.)
- Initial version of SEAL system abstraction library

- Initial version of SEAL utility library
- Proposal for external software decision process

? Later deliverables

- Incorporation of the CLHEP evolution decided with the CLHEP editors and LCG architects.

2 Component Model and Plug-in Manager

Develop or adapt a basic set of mechanisms and base classes for managing creation of objects (factories), lifetime, multiplicity and scope, component communication and interface discovery. Develop the plugging manager is responsible for loading, activating, deactivating and unloading plug-ins at run-time.

? Main activities and tasks

- Define component and interface model following the blueprint report guidance. This involves defining interfaces, abstract factories, etc.
- Develop plug-in Manager. Service in charge of managing, querying, [un]loading plug-ins and application bootstrapping (initialization)
- Define "Object management protocol" to define the object lifetime strategy.
- Document Component Model

? Proposed v1 deliverables

- Basic set of interfaces and base classes to support the *Component Model.*
- Initial version of plug-in Manager. This initial version should be sufficient to be used by the POOL project.
- Description of the Component Model and Object Management Protocol

? Later deliverables

- Plug-in Manager with sufficient functionality to be used by experiment frameworks

3 LCG Object Dictionary

Develop an object dictionary to provide reflection functionality by complementing the native C++ language features. The two aspects: population of the dictionary and reading the dictionary information through the reflection interface are considered

? Main activities and tasks

- Reflection packages (Reflection and ReflectionBuilder), currently part of the POOL project, should be imported and enhanced with some

improved functionality with respect to templated types and method stubs.

- Develop tools for populating dictionary from C++ header files, which is required by CMS and ATLAS. Investigate *gcc-xml* technology. This work has originally been carried out in the context of the POOL project.
- Develop gateway of the object dictionary to Python (Python binding). This work is interesting for completeness and as a usability exercise of the reflection interface provided by the dictionary.
- Develop gateway from ROOT to object dictionary. Be able to populate dictionary from CINT (inverse direction to the one developed currently in POOL). This should allow to interact to any ROOT object as if it was defined in the LCG dictionary

? Proposed v1 deliverables

- Reflection packages with small improvements: a) replace static *stub functions* by function objects, b) exploit templates for generation of *stub functions*
- Generation of dictionary from header files (partial C++ support) sufficient for CMS and ATLAS event model
- Python binding to the dictionary using *Boost.Python* technology while waiting for the result of the evaluation carried in the scripting support work package.

? Later deliverables

- Full C++ support for the generation of dictionary.
- Gateway from ROOT to object dictionary.

4 Basic Framework Services

Develop basic framework services for message reporting, exception handling, component configuration, “event” management, object “white board”, etc. Other services candidate to be developed will be identified by the other LCG applications area projects.

? Main activities and tasks

- Develop set of basic services for message reporting, exception handling, component configuration, “event” management, etc. It is understood that more services of common interest will be identified in other projects and they will also be developed.
- Develop object “whiteboard” as described in the Blueprint RTAG report. Study interaction of this component with the persistency services, visualization services and others.

? Proposed v1 deliverables

- Minimal set of basic services sufficient for POOL project: message reporting, exception handling, component configuration.

? Later deliverables

- More complete the list of basic services.

5 Scripting Services

Provide the basic infrastructure to support scripting. In particular, bindings for *Python* and *CINT* of the basic services will be needed to provide a "component bus" that allow easy integration of components, possibly implemented in a variety of languages, providing a wide variety of functionality.

? Main activities and tasks

- Evaluate existing the technologies for developing Python bindings (Python extension modules) of C++ classes (SWIG, Boost.Python, SIP,...) . Define guidelines for developing Python bindings and in particular study the way to solve the Python extension modules inter-dependencies.
- Develop Python bindings following the guidelines for the standard services and utility libraries developed in SEAL
- Enable scripting for application configuration
- Upgrade the existing package for binding ROOT to Python, which enables the interaction with any ROOT class from Python, following the defined guidelines.

? Proposed v1 deliverables

- Evaluation report of the technologies for creating Python bindings. Guidelines for creating Python bindings.
- ROOT python bindings (PyROOT) following guidelines.

? Later deliverables

- Bindings to all SEAL provided services and libraries

6 Grid Services

It is expected that the API to the Grid middleware services will be available as a set of linkable libraries. This work package should provide a common interface to the Grid services to be used by the other LCG application area projects as independent as possible of the middleware implementation.

? Main activities and tasks

- Gather requirements from POOL, PI for GRID-enabled services
- Provide common interface to various Grid middleware

? Proposed v1 deliverables

- None

7 Education and Documentation

Under this work package we group one of the main activities in the project. It is very important for the success of the project that the core libraries and services either developed in the project or supported third party ones are correctly documented and property advertised to the other LCG projects developers and to the experiment's framework developers which will need to integrate the experiment specific code with the provided LCG components.

? Tasks
  - Write the documentation of the software components developed by project.
  - Facilitate access to the documentation of third-party software components and presented in a coherent fashion with the rest of the documentation.
  - Develop tutorials presenting the project deliverables.
  - Help incorporating SEAL components into LCG projects and experiment frameworks.

? Proposed v1 deliverables
  - Documentation of all delivered components in version v1

3.2.2 Schedule Allocation

? **March 2003 – v1 alpha version**.
This version is defined as including all the essential components with their functionality sufficient for the needs of the other existing LCG projects (e.g. POOL).

? **June 2003 – v1 beta version**.
This version is defined as including all the essential components with their functionality sufficient to be adopted by experiments frameworks if they wish.

? **December 2003 – v2 version.**
This version is defined as the first complete version including implementations for all components known at the time of this plan.

3.2.3 Resource Allocation

Table 3 shows the estimates of the required FTE for each of the defined work packages. Concerning the currently available resources, the total number is correct but their sharing between the work packages is just a guess since we have not yet assign people to work packages at this phase of the project.

Table 3 Resource allocation

WBS	Name	FTE (available/required)
1	Foundation and Utility libraries	0.5 / 1.0
2	Component Model and Plug-in Manager	0.5 / 0.5
3	LCG Object Dictionary	0.5 / 1.5
4	Basic Framework Services	0.5 / 1.0
5	Scripting Services	0.5 / 1.0
6	Grid Services	0.0 / 1.5
7	Education and Documentation	0.5 / 1.5

3.3 Project Tracking Plan

3.3.1 Requirements Management

- After each public release of the software every 3 months, the requirements can be reviewed by the experiments. If changes are needed, the proposed modifications will be added into the *bug tracking system*[1].
- The impact of the proposed requirements changes will be assessed based on the product scope and quality, and on project schedule, budget, resources and risk factors.
- Using the *bug tracking system* we expect to be able to trace the changes in the requirements to the versions of the software implementing such changes.

3.3.2 Schedule Control

- The project has defined a number of major milestones.
- The progress of the work completed will be measured at the major and minor milestones.
- The project plan will be controlled and tracked quarterly.

3.3.3 Budget Control

- No special need to control the budget.

3.3.4 Quality Control

- To measure and control the quality of the work and the resulting work products will be done by making statistics of the number of defects of the software (bugs) using the *bug tracking system.*
- The quality control processes defined in collaboration with the SPI project will be applied to the project.

[1] GNU Savannah service provided by SPI

3.3.5 Reporting

? Specify the reporting mechanisms, report formats and information flows to be used in communicating the status of requirements, schedule, budget, quality, and other desired or required status metrics within the project and to entities external to the project.

? Specify the methods, tools and techniques of communication.

? Specify a frequency and detail of communications related to project management and metrics measurement that is consistent with the project scope, criticality, risk and visibility.

3.3.6 Project Metrics

? We still need to define the metrics which would provide meaningful measures of the quality, usability, maintainability, modularity, etc. of the foundations libraries and basic framework services.

? The frequency for collecting metrics data is defined to be at the same frequency as the public releases. Every three months.

3.4 Risk Management Plan

We have not defined a risk management plan yet. This plan should identify, analyze, and prioritize the project risk factors. We need describe the impact of the following risk factors:

- risks in the customer-project relationship,
- technological risks,
- risks caused by the size and complexity of the product,
- risks in the development and target environments,
- risks in personnel acquisition, skill levels and retention
- risks to schedule and budget, and
- risks in achieving customer acceptance of the deliverables.

3.5 Project Closeout Plan

We have not defined a project closeout plan yet.

4. Technical Process Plans

4.1 Process Model

- Define the relationships among major project work activities and supporting processes.
- Describe the flow of information and work products among activities and functions.
- Specify the timing of work products to be generated.
- Identify the reviews to be conducted.
- Specify the major milestones to be achieved.
- Define the baselines to be established.
- Identify the project deliverable to be completed.
- Specify the required approvals within the duration of the project.
- In the process model for the project, include project initiation and project termination activities.
- Use a combination of graphical and textual notations to describe the project process model.
- Indicate any tailoring of your organization's standard process model for a project.

4.2 Methods, Tools, and Techniques

- Specify the development methodologies, programming languages and other notations, and the processes, tools and techniques to be used to specify, design, build, test, integrate, document, deliver, modify and maintain the project deliverable and non-deliverable work products.
- Specify the technical standards, policies, and procedures governing development and/or modification of the work products.

4.3 Infrastructure

- Specify the plan for establishing and maintaining the development environment (hardware, operating system, network and software), and the policies, procedures, standards, and facilities required to conduct the IM/IT project. These resources may include workstations, local area networks, software tools for analysis, design implementations, testing, and project management, desks, office space, and provisions for physical security, administrative personnel, and janitorial services.

4.4 Product Acceptance

- Specify the plan for customer acceptance of the deliverables generated by the IM/IT project.
- Specify objective criteria for determining acceptability of the deliverables.
- Reference a formal agreement of the acceptance criteria signed by representatives of theIM/ITorganization and the customer.
- Specify any technical processes, methods, or tools required for deliverable acceptance, such as testing, demonstration, analysis and inspection.

5. Supporting Process Plans

5.1 Configuration Management

? Specify or reference the configuration management plan for the IM/IT project, providing the information identified in the following lines.

? Specify the methods that will be used to perform the following activities:

- configuration identification,
- configuration control,
- status accounting,
- evaluation, and
- release management.

? Specify the processes of configuration management including procedures for the following activities:

- initial baselining of work products,
- logging and analysis of change requests,
- change control board procedures,
- tracking of changes in progress, and
- procedures for notification of concerned parties when baselines are established or changed.

? Identify the automated configuration management tools used to support the configuration management process.

5.2 Verification and Validation

? Specify or reference the verification and validation plan for the IM/IT project, providing the information identified in the following lines.

? Specify the scope, tools, techniques and responsibilities for the verification and validation work activities.

? Specify the organizational relationships and degrees of independence between development activities and verification and validation activities.

? Specify the use of verification techniques such as traceability, milestone reviews, progress reviews, peer reviews, prototyping, simulation and modeling.

? Specify the use of validation techniques such as testing, demonstration, analysis and inspection.

? Identify the automated tools to be used in verification and validation.

5.3 Documentation

? Specify the plans for generating non-deliverable and deliverable project documentation.

? Specify the organizational entities responsible for providing input information, and for generating and reviewing the project documentation.

? Specify the following information or object identification:

- list of documents to be prepared,
- controlling template or standard for each document,
- who will prepare each document,
- who will review each document,
- due dates for review copies,
- due dates for initial baseline versions, and
- a distribution list for review copies and baseline versions and quantities required

5.4 Quality Assurance

? Specify or reference the quality assurance plan for the IM/IT project, containing the information identified in the following lines.

? Specify the plans for assuring that the IM/IT project fulfills its commitments to the IM/IT process and the IM/IT product as specified in the requirements specification, the IM/IT Project Management Plan, supporting plans and any standards, procedures, or guidelines to which the process or the product must adhere.

? As applicable, specify the quality assurance procedures to be used, such as analysis, inspection, review, audit, and assessment.

? Indicate the relationship among the quality assurance, verification and validation, review, audit, configuration management, system engineering, and assessment processes.

5.5 Reviews and Audits

? Specify the schedule, resources, and processes, and procedures to be used in conducting project reviews and audits.

? Specify the plans for joint customer-project reviews, management progress reviews, developer peer reviews, quality assurance audits, and customer-conducted reviews and audits.

? List the external agencies that approve or regulate any project deliverable.

5.6 Problem Resolution

- Specify the resources, methods, tools, techniques and procedures to be used in reporting, analyzing, prioritizing and processing IM/IT problem reports generated during the project.
- Indicate the roles of development, configuration management, the change control board, and verification and validation in problem resolution work activities.
- Provide for separate tracking of effort expended on problem reporting, analysis and resolution, so that rework can be tracked and process improvement accomplished.

5.7 Process Improvement

- Specify the plans for periodically assessing the project, for determining areas for improvement, and for implementing the improvement plans.
- Ensure that the process improvement plan is closely related to the problem resolution plan.
- Include in the improvement plan, a process to identify the project processes that can be improved without serious disruption to an ongoing project, and to identify the project processes that can best be improved by process improvement initiatives at the organizational level.