SAGA
new frontier group

METHODOLOGY, PM AND TECHNICAL PROPOSAL

For General Purposes

Saga Company
Address: **64a Zorana Đinđića Blvd. | 11070 Belgrade, Serbia**
URL**: www.saga.rs**

office@saga.rs

CONTENTS

General principles

Our general principles we are following in our projects are strongly focussed to open cooperation with the client and to a perception of its needs.

Partnership with the Client

We are available to the client on a daily basis for accepting of the requirements and we are in the position to give assistance to the client. Our long-term experiences and knowledge are prepared for achieving successful results and goals in favour for the client. Client's interests are our priorities and the client's satisfaction is the key issue and measure for our success.

Understandable project outputs

Documents we are providing on planed business functionality of the system are written with awareness of business domain terminology – not requiring any IT specific technical knowledge. We are expecting that their readers are customs procedural experts, not IT experts. It is the key prerequisite for the mutual understanding leading to efficient cooperation and predictable results with compliance to client's expectations. This business documentation is later elaborated into standard analytical documents of system analysis and design using standard notations and methodologies – primarily on our responsibility. We will never require client's approval on low technical design documents like class diagrams and so on requiring deep IT specific knowledge.

Complex approach to project goal

In projects like this, a complex solution balancing the needs of all participated subjects is necessary. There must be a complex and internally consistent design of all process details and system components.

Experience – worldwide and regional

With its years of experience in managing international IT projects, Saga Company has developed an in-depth expertise in IT risk management. Based on its assets, our company will make sure that throughout the project timeframe risk management is assessed and dealt by establishing a close working relationship with the Beneficiary staff and managers. Risk management is an inherent part of our proposed methodology to be applied for this project.

Team formation

Saga Company has invested quite some efforts to create a team that will ensure the project with successful project completion.

Here are some key differentiators:

- A good synergy based on motivation, teamwork, information sharing will be part of our project organisation.
- The project team and all key and non-key experts have proven ability to successfully manage the various aspects of the project.
- The team experience on other government projects aspects, such as financial controlling and budgeting, is strong.
- Saga Company has an in-depth IT experience in a several projects recently conducted for public sector in Serbia. It will bring his broad project knowledge to this contract. To be highlighted also, is the fact that it has a clear understanding of governmental projects with operational aspects and with experience in IT design.
- Good and careful engineers, knowledgeable in different dimensions of ICT and having an international experience, are assigned to the team in order to be able to deliver outstanding results.
- Aside the core team, Saga Company has a pool of professionals who will support the project as short-term Non-Key experts on various topics.

Project Methodology

A methodology is a system of practices, techniques, procedures and rules used by those who work in a discipline.

Saga is ISO IEC 27000 certified.

For a number of years, our professional services have been realised using a mainstream and successful methodology and software development life cycle process based on ISO/IEC 12207.

This International Standard ISO/IEC 12207 establishes a common framework for software life cycle processes, with well-defined terminology, that can be referenced by the software industry. It contains processes, activities, and tasks that are to be applied during the acquisition of a system that contains software, a stand-alone software product, and software service and during the supple, development, operation, and maintenance of software products. Software includes the software portion of firmware.

For each customer, the methodology is applied by selecting rigorously the processes of the project.

At the security side as ISO IEC 27000 certified company we are compliant with the ISO/IEC 17799 standard for security as (or not) required by the ToR.

Within the context of the following project, Saga will strictly deliver services like:

- Technical assistance
- Technical and User documentation
- Training

In order to fulfill the project requirements, we have prepared a methodological approach based on ISO/IEC 12207 and ISO/IEC 17799:2005.

Broader understanding of project methodology would mean the best practice frameworks we put together to get projects done.

Relation to Other Systems/Project

Beneficiary is due to timely and proactively inform the Supplier of eventual other projects / activities that may affect the Project and are not being a integral part of the ToR. This is necessary if the timely finalization of the Project Objective is the must.

Process Groups

Agile Unified Process describes a simple, easy to understand approach to developing business application software using agile techniques and concepts. It is based on five process groups.

Initiating Process Group

The Initiating Process Group consists of those processes performed to define a new project or a new phase of an existing project by obtaining authorization to start the project or phase. Within the initiating processes, the initial scope is defined and initial financial resources are committed. Internal and external stakeholders who will interact and influence the overall outcome of the project are identified.

Planning Process Group

The Planning Process Group consists of those processes performed to establish the total scope of the effort, define and refine the objectives, and develop the course of action required to attain those objectives. The planning processes develop the project management plan and the project documents that will be used to carry out the project. The multi-dimensional nature of project management creates repeated feedback loops for additional analysis. As more project information or characteristics are gathered and understood, additional planning may be required. The project management plan and project documents developed as outputs from the Planning Process Group will explore all aspects of the scope, time, costs, quality, communication, risk and procurements.

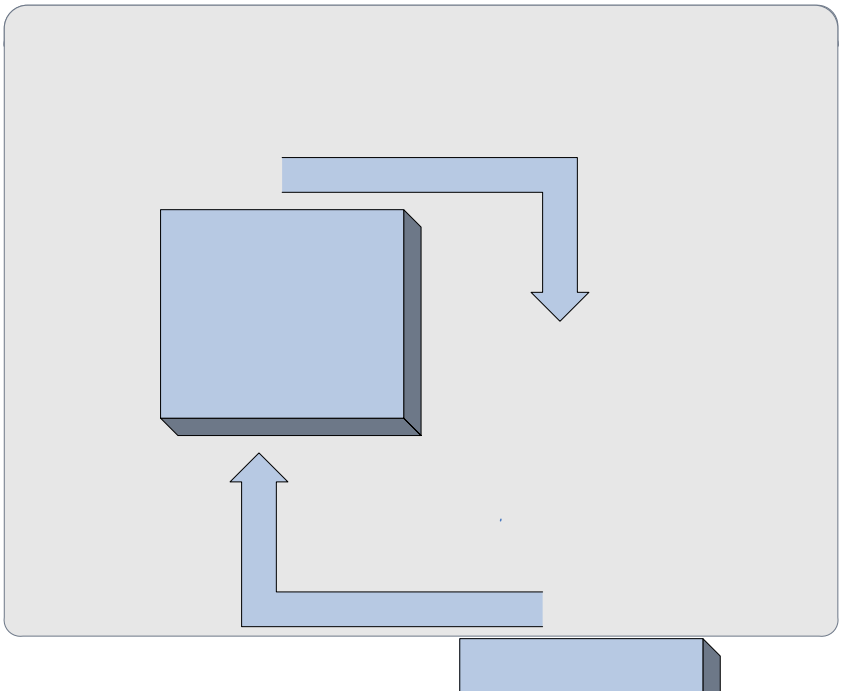


Figure 1 – Monitoring and Controlling Process Groups

Executing Process Group

The Executing Process Group consists of those processes performed to complete the work defined in the project management plan to satisfy the project specifications. This Process Group involves coordinating people and resources, as well as integrating and performing the activities of the project in accordance with the project management plan.

Monitoring and Controlling Process Group

The Monitoring and Controlling Process Group consists of those processes required to track, review and regulate the progress and performance of the project; identify any areas in which changes to the plan are required; and initiate the corresponding changes. The key benefit of this Process Group is that project performance is observed and measured regularly and consistently to identify variances from the project management plan. The Monitoring and Controlling Process Group also includes:

- Controlling changes and recommending preventive action in anticipation of possible problems
- Monitoring the ongoing project activities against the project management plan and the project performance baseline, and
- Influencing the factors that could circumvent integrated change control so only approved changes are implemented.

Closing Process Group

The Closing Process Group consists of those processes performed to finalize all activities across all Project Management Process Groups to formally complete the project, phase, or contractual obligations. This process Group, when completed, verifies that the defined processes are completed within all the Process Groups to close the project or project phase, as appropriate, and formally establishes that the project or project phase is complete.

General Description of the Implementation Methodology

According to above mentioned Project Process Group, processes involved into implementation will be performed in accordance to the Agile Unified Process (AUP) and Bidder procedures which represent the best practice of ISO 9001:2008 norm.

Agine Unified Process (AUP) provides an outline of all the tasks to be considered in the life-cycle of an application implementation and can be tailored to reflect the specific requirements of every project.

AUP includes specific instructions which describe for the implementation consultant what to do, who should do it, why it should be done, when to do it, and how to do it. AUP also includes deliverable templates which act as a checklist of contents and provide a standard for layout and format of the deliverables.

Agile Unified Process (AUP) describes a simple, easy to understand approach to developing business application software using agile techniques and concepts yet still remaining true to the Rational Unified Process. The AUP applies agile techniques including test driven development (TDD), Agile Modeling, agile change management, and database refactoring to improve productivity. The UP is a software product engineering process framework which may be approached using three perspectives, including collaborations, context, and interactions which focus on a lifecycle composed of phases, disciplines, and iterations.

Figure 2 shows a conceptual view of the essential elements that constitute the UP and AUP.

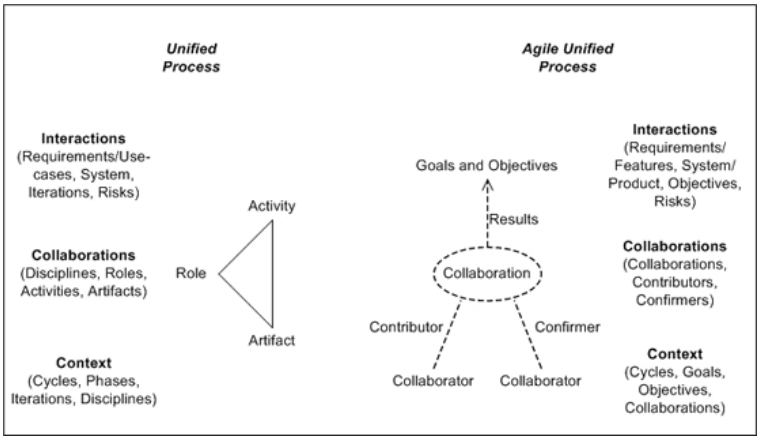


Figure 2 - The UP and AUP

A collaboration involves an interaction within a context. In the UP, a collaboration captures who does what activities (how) on what work products. Thus, it establishes the elements of a project. Collaborations involve workers (roles), activities, and work products (artifacts). In the AUP, collaborations focus on collaborators (contributors and confirmers), goals and objectives, and results.

A context emphasizes the structural or static aspect of a collaboration, the elements that collaborate and their conglomeration or spatial relationships. In the UP, a context captures when and where such activities should be done and work products (artifacts) produced and consumed. Thus, it establishes the context for a project. Contexts involve development cycles and phases, iterations, and disciplines. The UP's phases include Inception, Elaboration, Construction, and Transition. The UP's disciplines include Business Modeling, Requirements, Analysis & Design, Implementation, Test, Deployment, Configuration & Change Management, Project Management, and Environment. In the AUP, contexts focus on goals.

An interaction emphasizes the behavioral or dynamic aspect of a collaboration, the elements that collaborate and their cooperation or temporal communication. In the UP, an interaction captures when and why such activities should be done and work products (artifacts) produced and consumed. Thus, it establishes the execution of a project. Interactions involve requirements or use-cases, a system and its architecture, iterations, and risk. In the AUP, interactions focus on objectives.

The Agile UP lifecycle is serial in the large, iterative in the small, delivering incremental releases over time. Model is an important part of the AUP, as can be seen, but it doesn't dominate the process - we want to stay agile by creating models and documents which are just barely good enough. Second, the Configuration and Change Management discipline is now the Configuration Management discipline. In agile development the change management activities are typically part of the requirements management efforts, which is part of the Model discipline.

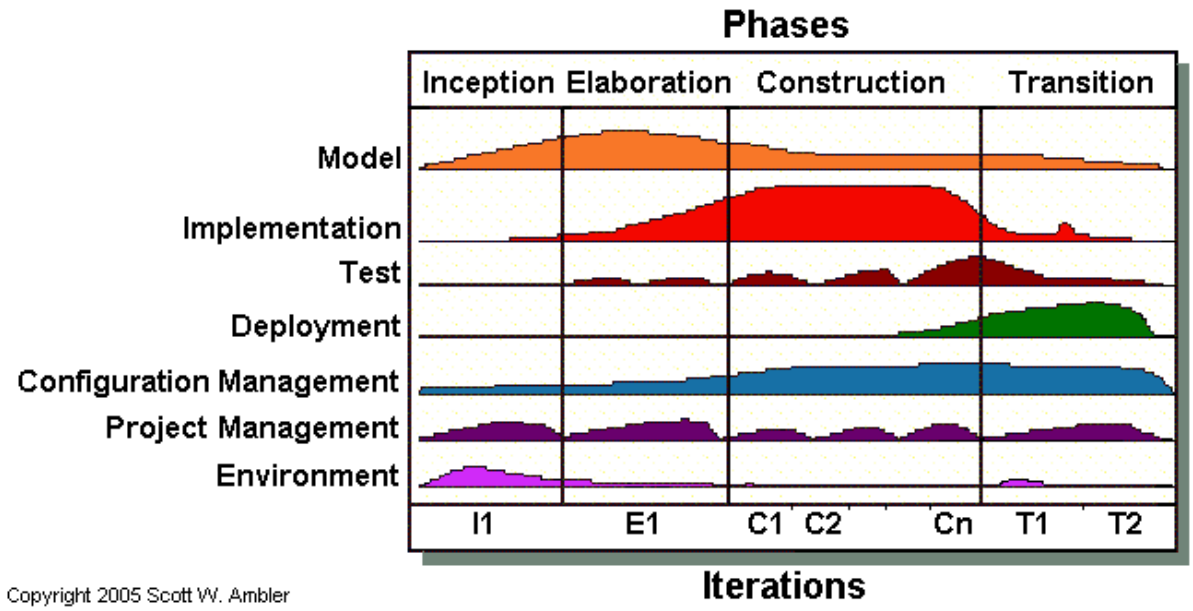


Figure 3 - Lifecycle of the AUP

Agile Unified Process Phases

Inception Phase

This phase shall be a highly productive period and we have therefore set ourselves a challenging number of tasks.

Mobilisation

The Mobilisation phase includes following activities:

- Establish the management structure of the project (activating the Project management Board).
- Mobilise the experts, assign tasks and prepare immediate work schedules.
- Finalise administrative and contractual arrangements with the Contracting Authority.
- Define administrative systems and processes for the management of project activities.

Project launch

During this period we undertake to:

- Revise the project implementation and management strategy outlined in this document to reflect the latest situation on the ground, prepare a revised resource utilisation and results chart; design an updated Project Log frame, draft an indicative annual Work Programme for the Project and a detailed Work Plan for the next reporting period.
- Carry out a review of the existing project documentation.
- Review the planning to take into account the last developments of the situation.
- Kick-off meeting.
- Draft the Inception Report.

The Inception report will contain analysis of existing situation; update the methodology of the project and its components. Set out detailed plan of work for the project indicating each activity, area of expertise and list of deliverables and identify the support personnel required.

The primary goals of the Inception phase are to achieve stakeholder consensus regarding the objectives for the project and to obtain funding. The major activities of the phase include:

1. **Define project scope**. This includes defining, at a high level, what the system will do. Equally important is to define what the system will not do. This establishes the boundaries within which the team will operate. This usually tales the form of a list of high-level features and/or point form use cases.
2. **Estimate cost and schedule**. At a high level, the schedule and cost for the project are estimated. General estimates are used for iterations in later phases, more specificity is used for the early iterations in Elaboration. This should not be construed as meaning that the entire project is

planned out at this point. As with all planning, those tasks that are to be completed in the near future are detailed more accurately and with greater confidence while tasks further down the line are understood to be estimates with larger margins of error. It has (finally) been recognized by most in the industry that it is not possible to schedule an entire project at its kick-off with any acceptable degree of confidence. The best that can be done is to plan for the near term accurately and the longer term as best as we can.

3. **Define risks**. The risks to the project are first defined here. Risk management is important an AUP project. The list of risks is a living compilation that will change over time as risks are identified, mitigated, avoided and/or materialize and dealt with. Risks drive the management of the project, as the highest priority risks drive the scheduling of iterations. Higher priority risks, for example, are addressed in earlier iterations than lower priority risks.
4. **Determine project feasibility.** The project must make sense from technical, operational, and business perspectives. In other words, we should be able to build it, once it's deployed we should be able to run it, and it should make economic sense to do these things. If the project isn't feasible, it should be cancelled.
5. **Prepare the project** environment. This includes reserving workspace for the team, requesting the people that will be needed, obtaining hardware and software that are needed immediately, and compiling a list of anticipated hardware and software that will be needed later. In addition the Team will tailor the AUP to meet The team's exact needs.

To exit the Inception phase the team must pass the Lifecycle Objectives (LCO) milestone. The main issues are whether the team understands the scope of the effort sufficiently and whether the stakeholders wish to fund the project. If the team passes this milestone the project moves to the Elaboration phase, otherwise the project may be re-directed or cancelled outright.

Inception Phase – How to Work By Disciples

Discipline	**Major Activities**
Model	• Initial, high-level requirements modeling • Initial, high-level architectural modeling
Implementation	• Technical prototyping • User interface prototyping
Test	• Initial test planning • Review initial project management work products • Review initial models
Deployment	• Identify the potential release window • Begin high-level deployment planning
Configuration Management	• Setup the configuration environment • Put all work products under CM control

Project Management	• Start building the team • Build relationships with The project stakeholders • Determine project feasibility • Develop a high-level schedule for the entire project • Develop a detailed iteration plan for the upcoming iteration • Manage risk • Obtain stakeholder support and funding • Close out the phase
Environment	• Set up the working environment • Identify the project category

Realization Phase – Elaboration sub-phase

As a part The primary goal of the Elaboration sub-phase is to prove the architecture for the system to be developed. The point is to ensure that the team can actually develop a system that satisfies the requirements, and the best way to do that is to build a end-to-end, working skeleton of the system called an "architectural prototype". This is actually a poor term for the concept because many people think that we throw prototypes away. Instead, The goal is to write high-quality, working software which meets several high risk (from a technical point of view) use cases to show that the system is technically feasible.

It is important to note that the requirements are not specified completely at this point. They are detailed only enough to understand architectural risks and to ensure that there is an understanding of the scope of each requirement so that subsequent planning can be carried out. Architectural risks are identified and prioritized; the significant ones are addressed during Elaboration. Addressing architectural risks may take several forms: research into similar system(s), a stand-alone test suite, a working prototype, etc. In most cases, a working prototype showcasing the architecture is completed. The system level architecture should also reflect The overall enterprise architecture.

During Elaboration, the team is also preparing for the Construction phase. As the team gains a handle on the architecture of the system, they begin setting up the environment for Construction by purchasing hardware, software, and tools. From a project management point of view, staffing is addressed; resources are requested and/or hired. Plans for communication and collaborating are finalized (especially important if the team is to be physically distributed).

To exit the Elaboration phase the team must pass the Lifecycle Architecture (LCA) milestone. The primary issues addressed with this milestone is whether the team has shown that they have a working end-to-end prototype which shows the team has a viable strategy to build the system and that the stakeholders are prepared to continue funding the project. If the team passes this milestone the project moves to the Construction phase, otherwise the project may be re-directed or cancelled.

Realization Phase, Elaboration sub-phase – How to Work By Discipline

Discipline	Major Activities
Model	• Identify technical risks • Architectural modeling • User interface prototyping
Implementation	• Prove the architecture
Test	• Validate the architecture • Evolve The test model
Deployment	• Update The deployment plan
Configuration Management	• Put all work products under CM control
Project Management	• Build the team • Protect the team • Obtain resources • Manage risk • Update The project plan • Close out the phase
Environment	• Evolve the working environment • Tailor the process materials

Realization Phase – Construction sub-phase

The focus of the Construction phase is to develop the system to the point where it is ready for pre-production testing. In previous phases, the majority of the requirements have been identified and the architecture for the system has been base-lined. Emphasis shifts now to prioritizing and understanding the requirements, model storming a solution, and then coding and testing the software. If necessary, early releases of the system are deployed, either internally or externally, to obtain user feedback.

To exit the Construction phase The team must pass the Initial Operational Capability (IOC) milestone. The primary issue here is whether the current version of the system is ready to move into The pre-production test environment for system and acceptance testing. If the team passes this milestone the project moves to the Transition phase, otherwise it may be re-directed or cancelled.

Realization Phase, Construction sub-phase – How to Work By Discipline

Discipline	Major Activities
Model	• Analysis model storming • Design model storming • Document critical design decisions
Implementation	• Test first • Build continuously • Evolve the domain logic • Evolve the user interface • Evolve the data schema • Develop interfaces to legacy assets • Write data conversion scripts
Test	• Test the software • Evolve The test model
Deployment	• Develop (de)installation scripts • Develop release notes • Develop initial documentation • Update The plan • Deploy the system into pre-production environments
Configuration Management	• Put all work products under CM control
Project Management	• Manage the team • Manage risk • Update The project plan • Close out the phase
Environment	• Support the team • Evolve the working environment • Setup training environment

Finalization/Transition Phase

The Finalization/Transition phase focuses on delivering the system into production. There may be extensive testing that takes place during this phase, including beta testing. Fine-tuning of the product takes place here as well as rework to address significant defects (the stakeholders may choose to accept the existence of some known defects in the current release).

The time and effort spent in Finalization/Transition varies from project to project. Shrink-wrapped software entails the manufacturing and distribution of software and documentation. Internal systems are generally simpler to deploy than external systems. High visibility systems may require extensive beta testing by small groups before release to the larger population. The release of a brand new system may entail hardware purchase and setup while updating an existing system may entail data conversions and extensive coordination with the user community. Every project is different.

To exit the Transition phase the team must pass the Product Release (PR) milestone. The primary issue here is whether the system can be safely and effectively deployed into production. If the team passes this milestone the project moves to production. If the project fails in any area above, the project may be re-directed or cancelled.

Finalization/Transition Phase – How to Work By Discipline

Discipline	**Major Activities**
Model	• Model storming • Finalize system overview documentation
Implementation	• Fix defects
Test	• Validate the system • Validate the documentation • Finalize the test model
Deployment	• Finalize the deployment package • Finalize documentation • Announce the deployment • Train people • Deploy the system into production
Configuration Management	• Put all work products under CM control
Project Management	• Manage the team • Close out the phase • Initiate the next project cycle

Environment	• Setup operations and/or support environments • Recover software licenses

Agile Unified Process Disciples

Disciplines are performed in an iterative manner, defining the activities which development team members perform to build, validate, and deliver working software which meets the needs of their stakeholders. The disciplines are:

- Model
- Implementation
- Test
- Deployment
- Configuration Management
- Project Management
- Environment

Model

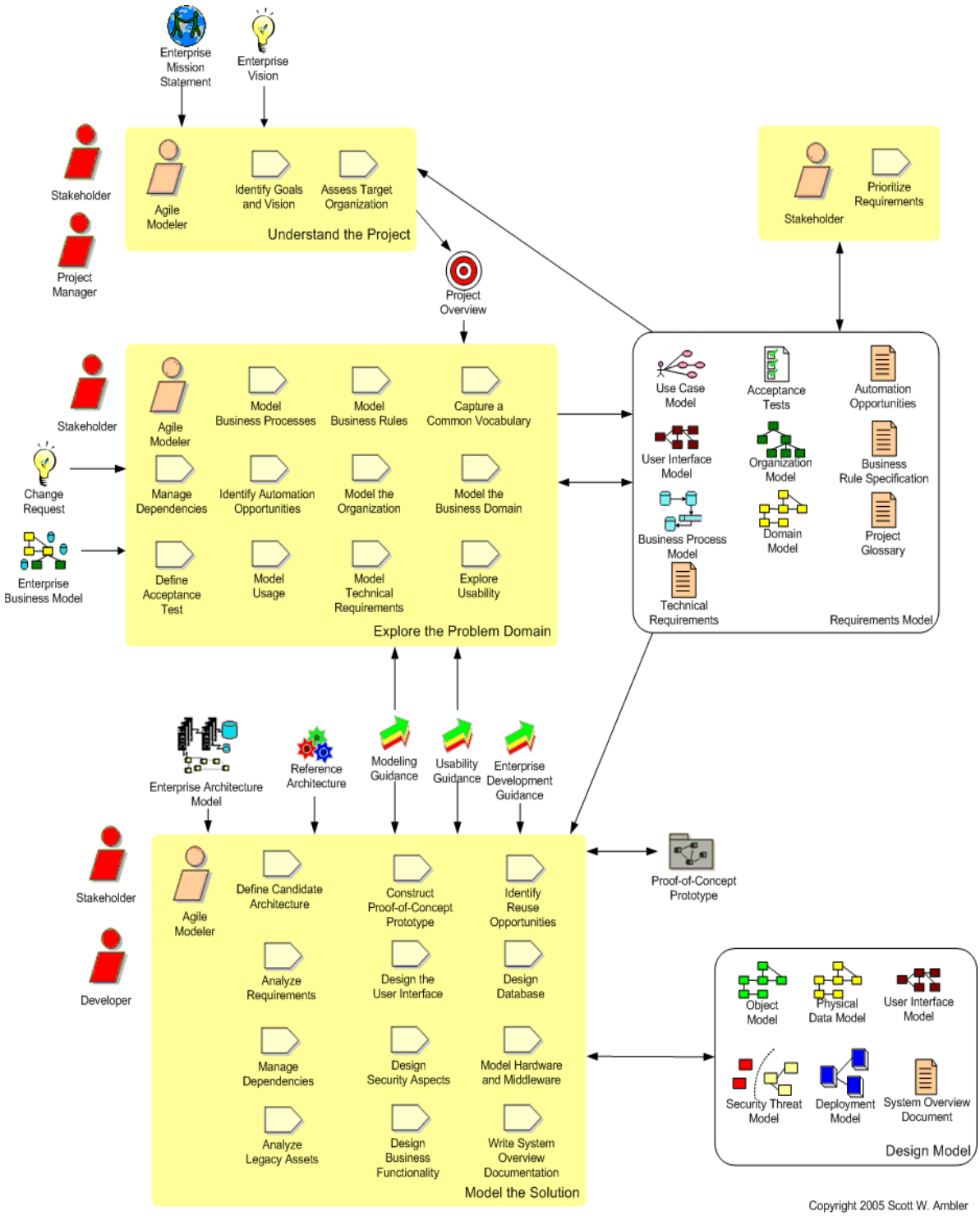


Figure 4 - The Agile Model Driven Development (AMDD) lifecycle (modified)

Model Disciple Phase by Phase

Phase	Activities
Inception	Initial, high-level requirements modeling (see Figure 1). The stakeholders should actively participate in the development of a high-level requirements model which defines the initial scope for the project and provides sufficient information for a rough estimate. We should consider: Exploring usage by writing point-form use cases. Identifying the business process(es) via the creation of data flow diagrams (DFDs). Identifying the major business entities and their relationships by working on a slim domain model. Identifying major business rules and technical requirements. For now, the name of the business entities, rules, and technical requirements are likely sufficient (we can get the details through model storming during the Construction phase). Starting development of a glossary describing important business and technical terms. Understanding the political structure within the stakeholder community via organizational modeling. Treating requirements like a prioritized stack which evolve over time (this supports true change management, not change prevention). Use cases, business rules, and technical requirements all belong on the stack. Initial, high-level architectural modeling.The goal is to identify a viable architectural strategy, critical input into the project planning activity as well as into the implementation efforts. The best way to work is to get several technical people, including some if not all of the developers, together in a room to develop an architectural strategy as you're talking around a plain-old-whiteboard (POW) creating free-form diagrams and perhaps some form of initial deployment model.
Elaboration	Identify technical risks. The requirements work products, in particular the use cases and technical requirements, will reveal potential technical risks to the project. These risks may include the introduction of new technology to the organization, a new use of existing technologies, significant load/stress on the application, or interfacing to

	external/legacy systems. The highest priority risks should be addressed by the implementation efforts in the development of an end-to-end skeleton of the system. Architectural modeling. As we build the architectural prototype we need to model storm some details to think through portions of the architecture. User interface prototyping. In parallel to the development of the architectural prototype we should also consider user interface prototyping of several major screens. The Team doesn't want to do too much prototyping because the requirements are likely to change and therefore the work will need to be discarded. The goal at the time should be to understand the main screens/pages of the user interface, with the understanding that they will change during Construction, and to identify the basic "look and feel" of the system.
Construction	Analysis model storming. During Construction iterations we will need to work closely with the project stakeholders to understand their needs on a just-in-time (JIT) basis. Important issues: Active stakeholder participation and inclusive modeling which use simple tools and techniques are critical to the success. The Project may require to flesh out the details of the use cases, perhaps visually using flow charts or UML activity diagrams instead of text descriptions. Explore the business rules and technical requirements in the same manner. It may need to interface to legacy assets such as existing systems or databases. Legacy analysis can be difficult and "politically charged" work. Instead of use case descriptions, business rule descriptions, and technical requirement descriptions we may find it more effective to simply write acceptance test cases instead. This enables us to come close to single sourcing information because we don't need to capture the requirement in both a requirements document and a test description. Because the user interface is the system to many of the stakeholders, the Team will likely discover that they prefer to focus on what the screens and reports look like instead of the development of other work products, so be prepared to prototype. Keep the project glossary up to date if there is one.

	Design model storming. During Construction iterations the goal is to do just enough modeling to think through the design of a single requirement, or portion thereof, before implementing the requirement. Agile modelers model directly with developers, they don't simply hand off models to them, and often take on the role of developer. The Team will need to create: UML sequence diagrams. These diagrams depict the dynamic logic within the source code. They are part of the object model but are usually throwaways unless we have a good CASE tool with reverse engineering capabilities. Whiteboards are great tools to create new ones. Deployment model. The Team typically need to create some sort of "overview diagram" depicted the deployment/network architecture of the system. Free-form diagrams. Typically created on whiteboards and then erased when no longer needed. Free-form diagrams are likely the most common model drawn. System overview documents typically include several free-form diagrams. UML class diagrams. If you're going to do any class diagramming use a modeling tool which generates source code. The class diagrams should be based on the domain model (if any). Security threat model. If security issues are a concern then we should consider modeling them to help us to think through the potential threats as well as how to address those threats. Physical data models. This is likely the most important design model, one which we should consider using a CASE tool to develop and maintain over time, particularly a tool which generates DDL code. It is possible to take an agile approach to data modeling. Document critical design decisions. As we make design decisions we should consider recording any that don't seem obvious, or that we believe someone else in the future would really like to know, as a start to the system overview documentation.
Transition	Model storming. We will need to do some JIT modeling to try to understand the root cause of a defect. Finalize system overview documentation. The best time to finalize the system overview documentation is during this phase when the scope of the system has truly stabilized. Use the critical design decisions, if we documented them during Construction, as the base from which to build this document. Other important

	information that you'll want in this document is a summary of the scope of the system and critical architecture diagrams (now might be the time to turn those free-form whiteboard sketches into nice looking diagrams using a drawing tool).

Implementation

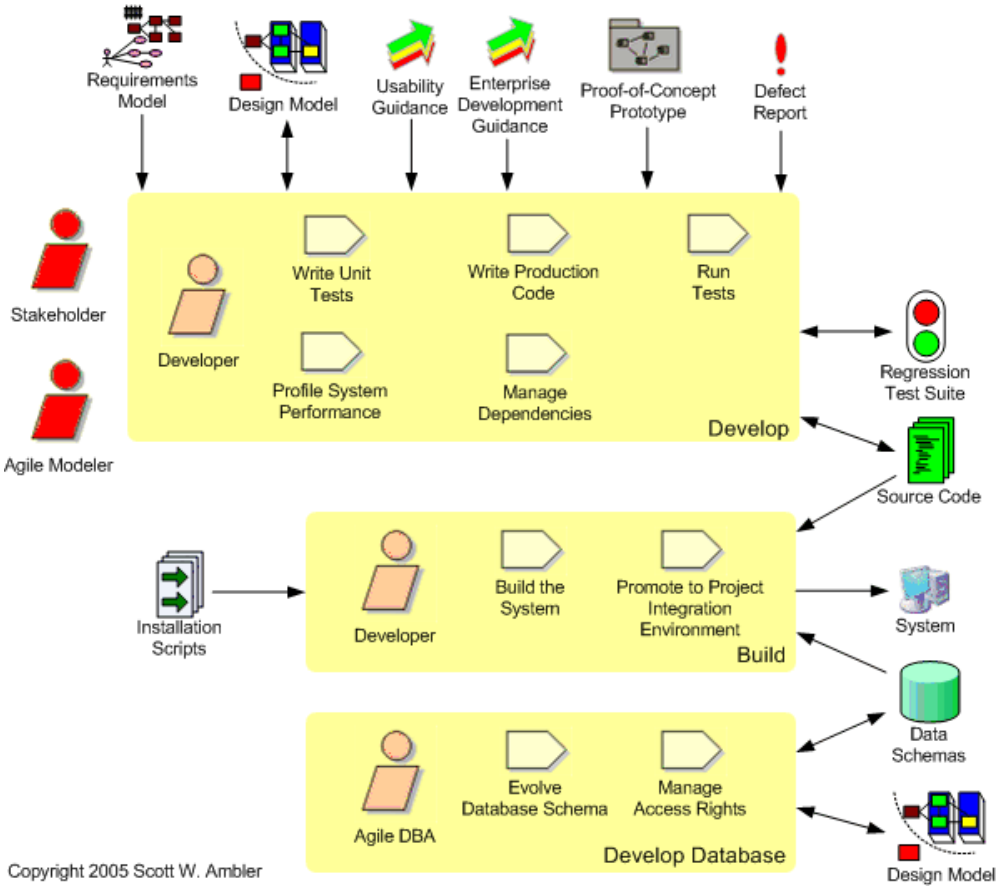


Figure 5 - Workflow

Implementation Disciple Phase by Phase

Phase	Activities
Inception	Technical prototyping. The Team might need to "spike" a small aspect of a requirement in order to understand it sufficiently, enabling us to estimate the effort required. These prototypes are typically small, "throw away" pieces of code.

	User interface prototyping. For most stakeholders the user interface (UI) -- the screens, reports, and manuals -- is the system. When the UI is potentially complex, or when the stakeholders want to see what they're going to get before they buy it, we will want to consider prototyping at least the main screens/pages. The UI prototype, usually a throw away at this point, would be used to convince the stakeholders that the Team understands their needs (which we would explore as part of the modeling efforts).
Elaboration	Prove the architecture. The critical activity within the Elaboration phase is to identify a potential architecture and then prove that the architecture works via the development of an end-to-end architectural prototype for the system, thereby mitigating much of the technical risk on the project. Technical prototypes such as this are production quality code that forms the foundation, or skeleton, of the system.
Construction	Test first. Take a TDD-based approach to all aspects of implementation. Build continuously. Daily builds are a good start, but ideally we want to build the system whenever the source code changes. Automate this using a product like Cruise Control which monitors the version control system for changes to the code and rebuilds as needed. Evolve the domain logic. Implement the business logic in the domain/business classes. Evolve the user interface. The user interface is the system to most of the users. Strive to make the software as usable as possible by following common usability and user interface design strategies. Evolve the data schema. The data schema should be evolved in step with the domain and user interface code. The Team will need to refactor the database just like we refactor the other types of code. Develop interfaces to legacy assets. We will often need to access existing functionality within legacy systems. This may be done via a variety of means, including a web services interface, a C-API, stored procedures, and so on. Legacy analysis is often an important part of the modeling efforts. Write data conversion scripts. We will need to access legacy data sources. These data sources often suffer from design and/or quality problems and as a result we will need to write data conversion scripts to put the data feed(s) into a useable format.
Transition	Fix defects. The focus shifts to fixing the defects found as a result of testing.

Testing

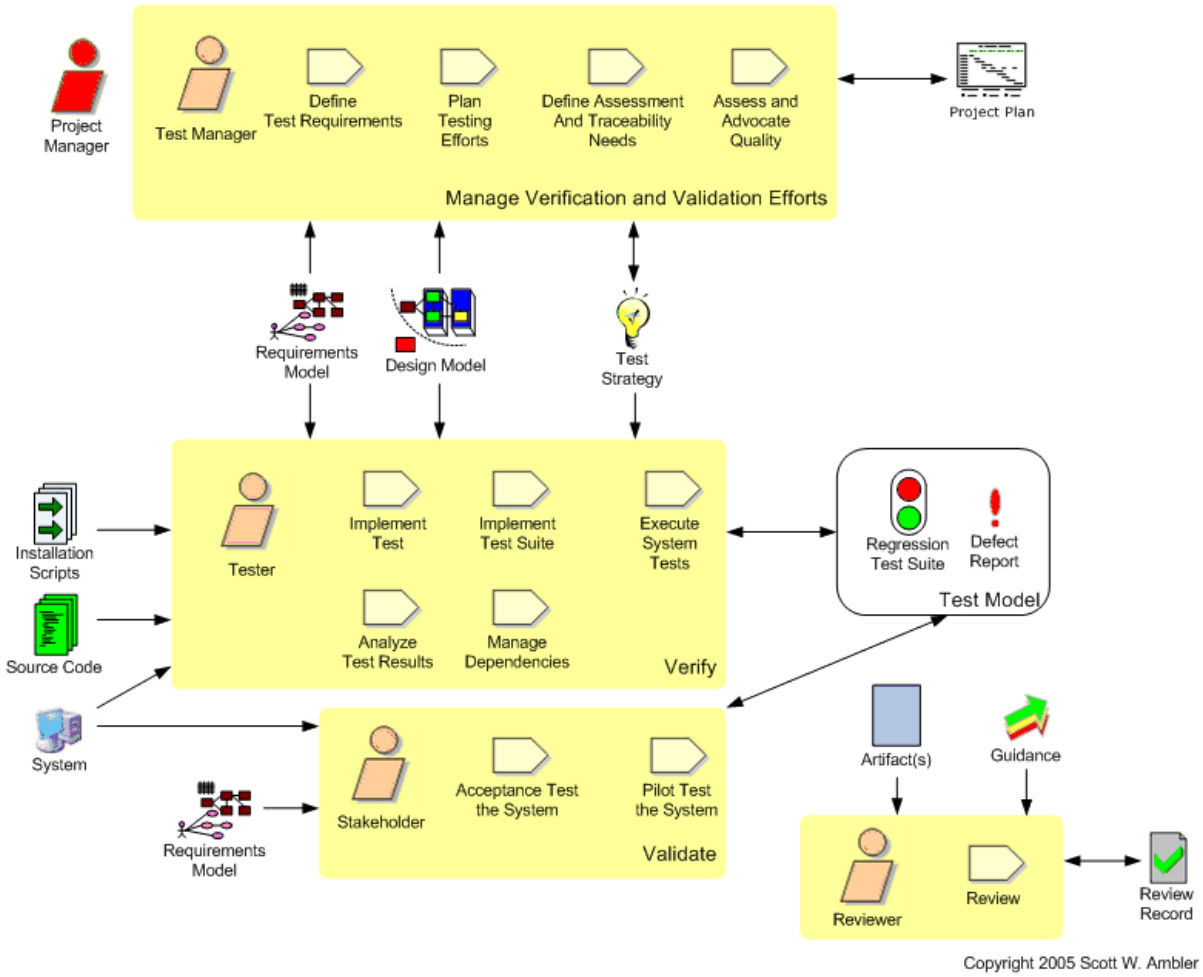


Figure 6 - Workflow

Test throughout the lifecycle. Next Figure depicts the Full Lifecycle Object Oriented Testing (FLOOT) lifecycle diagram.

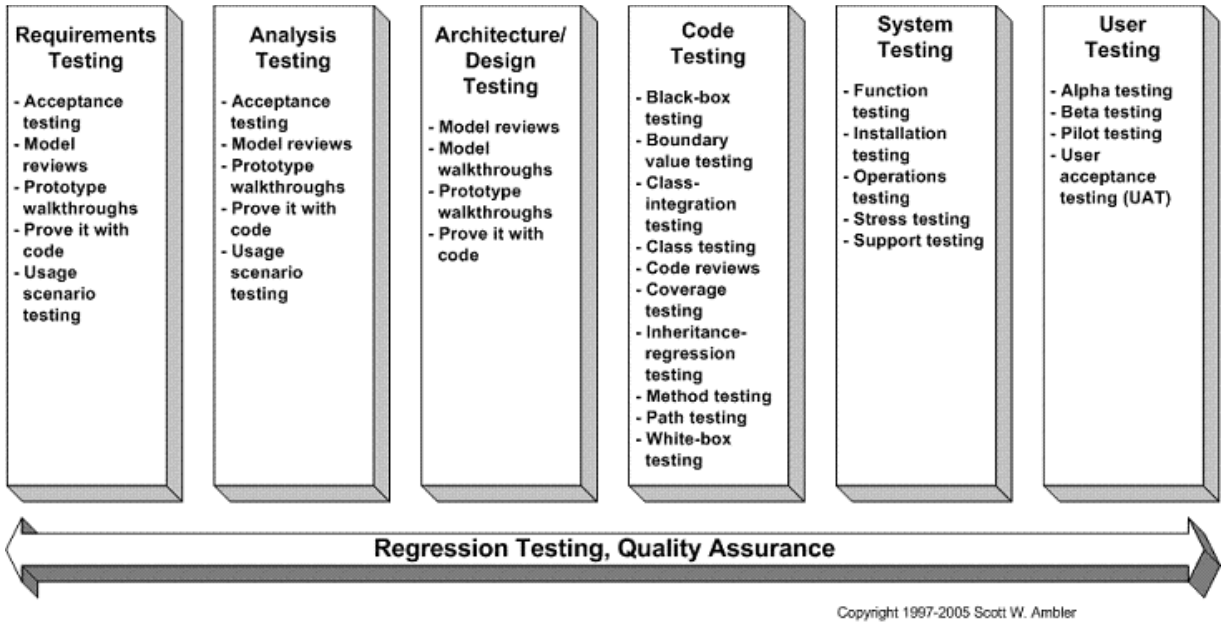


Figure 7 - The FLOOT lifecycle

Test Disciple Phase by Phase

Phase	Activities
Inception	Initial test planning. Should be very high-level at first. The main goal is to identify how much testing you'll need to do, who will be responsible for it, the level of stakeholder participation required, and the types of tools and environments required (an Environment discipline issue). Review initial project management work products. Towards the end of this phase the initial project plan, vision, and so on should be available. These work products are often reviewed, typically as part of the milestone review, by key project stakeholders. Review initial models. A high-level, initial requirements model, and perhaps even an initial architecture model, should be produced by the modeling efforts. We may choose to review this work with stakeholders, particularly if we want to communicate the scope and potential architecture of the system to a wider range of people than were actively involved in the development of the models.
Elaboration	Validate the architecture. The Team should take a test-driven development (TDD) approach to building the technical prototype which proves the architecture of the system. An important aspect of the milestone review is the validation of the architecture, which might be something as simple as presenting an overview of the architecture and the results of the prototyping efforts to the stakeholders. Or, it might be something as complex as a formal review of all the work during this phase.

	Evolve the test model. The team will develop a regression test suite, comprised of unit tests from the test driven development (TDD) efforts in implementation, the acceptance tests from the modeling efforts, and the system tests (e.g. for function, integration, load, ... testing). We may also need to maintain traceability between the requirements, tests, and source code to show how we have validated the implemented requirements. At this point the defect reports would simply be the output of the test suite.
Construction	Test the software. In addition to unit testing by developers we need to do installation testing of the deployment scripts, system testing efforts such as load/stress testing and function testing, and user acceptance testing. Because the software evolves throughout the projects so will the test suite. The more often we promote the code into a pre-production test environment, and then test it appropriately, the better the Transition phase testing activities will be. Evolve the test model. See above.
Transition	Validate the system. The focus will be on "testing in the large" activities such as system testing, integration testing, acceptance testing, and pilot/beta testing. The goal is to fully test the system within the pre-production testing environment(s). Validate the documentation. The system documentation (system overview, user, support, and operations documentation), and the training materials will need to be validated. This can be done in reviews or better yet as part of the pilot/beta testing. Finalize the test model. The Team will continue to run the regression test suite, and update it as needed, until the system is ready to be deployed into production. The defect reporting will likely become more formal, the found defects will likely be recorded, along with appropriate details, so that developers can implement fixes.

Deployment

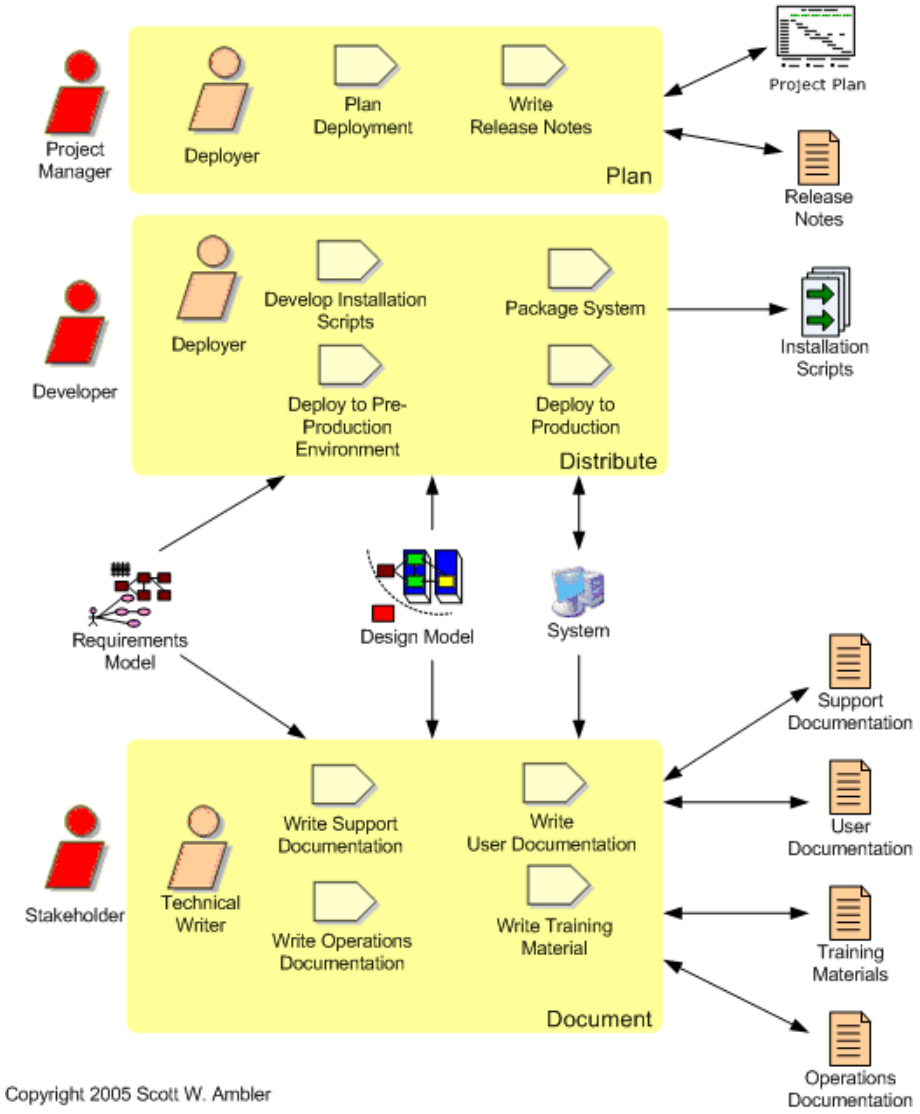


Figure 8 - Workflow

Deployment Disciple Phase by Phase

Phase	Activities
Inception	Identify the potential release window. For example, we may be required to deliver the software before the end of the year but after the release of another application that is planned to go into production at the end of September. Therefore the release window is between October 1st and December 31st, but we may decide to be conservative and set it to be between November 1st and December 15th to take into account deployment

	problems with the other project and the holidays at the end of December. Defining the release window early in the project helps we in the project planning efforts. Begin high-level deployment planning. This effort will focus on initial release planning for the system, identifying the deployment audience, and identifying the potential release window. The main goal should be to determine a general deployment strategy: based on the understanding of the project, does it make sense to release the software all at once or as phased releases? Because we won't have finalized the architectural strategy yet, we may decide to begin deployment planning during the Elaboration phase instead.
Elaboration	Update the deployment plan. An important part of defining the architecture is defining the deployment configurations for the system, perhaps we will support a three-tier client/server configuration for internal users connected to the network, an HTML-based interface for Internet usage, and a stand-alone single user version for disconnected users. Each individual deployment configuration could be documented with some form of deployment model that defines how the actual software components are organized and deployed to hardware components. Understanding the configurations will help us to identify the different types of installations you'll have to do, which in turn should provide insight into the overall deployment effort.
Construction	Develop (de)installation scripts. As we develop the system we should also write, and test, the installation and de-installation scripts required to deploy it into the pre-production testing environment. This scripts should be written so that we can simply reconfigure them to deploy into production. Develop release notes. The release notes should summarize the "good things to know" about the current release of the system which you're building. Point form notes are likely sufficient for now. Develop initial documentation. In addition to delivering working software, we will also need to deliver system documentation (operations, support, system overview, and user documentation), as well as the training materials. If you're creating new documents then we can at least begin to define their structure and add point form notes to them. The Team doesn't want to put much work into the documents now because the system is still evolving. Update the plan. As development of the system progresses, so should the deployment plan. We will likely find that we need to negotiate the deployment plan with the operations and support departments, as well as with other projects that are also planning on deploying software, so that the project fits into the overall corporate system deployment plan. Deploy the system into pre-production environments. We should regularly deploy the system into pre-production environments for QA/testing as well as for demonstrations to

	the stakeholders. The more experience we get doing this the easier the actual deployment into production is likely to be.
Transition	Finalize the deployment package. To finalize the packaging of the software we need to define its deployment baseline, a configuration management activity, and we need to perform a "final" build the software, an Implementation workflow task. Finalize documentation. The majority of the system documentation (operations, support, system overview, and user documentation) is typically written during this phase because the system's functionality has stabilized. Announce the deployment. The Team should announce the anticipated deployment schedule including the expected training and installation dates. The Team need to train the operations, support, and user communities as appropriate during this phase. Train people. Training the project customers – users, management, operations staff, and support staff – is always an important part of deployment. Remember that the customers may need training beyond that of learning how to work with the application. For example, this may be the first time that some users are working with a PC, a browser, or even a mouse. Similarly, this may be the first time that the operations staff is working with a new technology that the system users and therefore will need to be trained and educated in that technology to qualify them to work with the system. Deploy the system into production. At this point we will perform any required data conversion, which may be a one-time batch job or a gradual conversion of the data as it is needed by the users. We may decide to run the new system in parallel with the existing system for several weeks to ensure that it actually works in production. The Team may also choose to deploy the system to a subset of the user community, called a pilot release, to verify that it runs for the smaller group before we choose to "inflict" the system on everyone. We may also need to deploy a version of the software to the support department so they can simulate production problems when users contact them for help.

Configuration Management

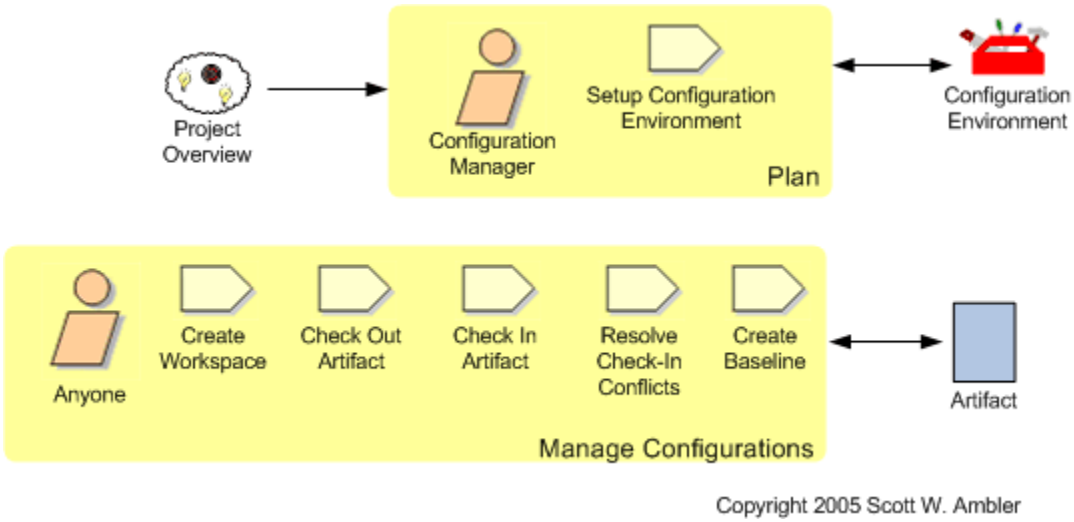


Figure 9 - Workflow

Configuration Management Disciple Phase by Phase

Phase	**Activities**
Inception	Setup the configuration environment. The Team need to do several things: • The CM repository will need to be installed if it is not already in place. • The appropriate folder/directory structure, which should follow any existing corporate guidance, needs to be created for the project team. • Project team members will need to be given access to the project folder(s) and any required client software installed on their workstations. • Project team members may also need to be trained in basic CM concepts and the tools. Put all work products under CM control. Everyone should put their work under CM control on a regular basis, checking it in/out as appropriate, resolving update conflicts as needed, and base-lining the work products when major versions are agreed upon.
Elaboration	Put all work products under CM control. See above.
Construction	Put all work products under CM control. See above.
Transition	Put all work products under CM control. See above.

Environment

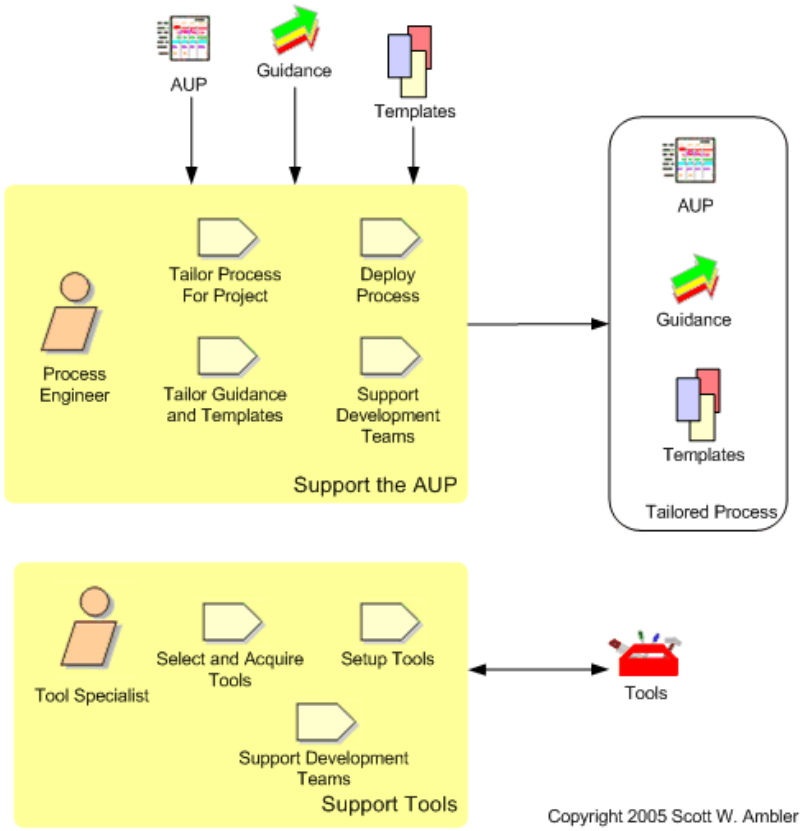


Figure 110 - Workflow

Environment Disciple Phase by Phase

Phase	Activities
Inception	Set up the working environment. Install workstations as needed as well as software for those workstations. This will be an ongoing task as people are added to the team over time. Identify the project category. Many organizations develop several base versions of their software process, for example one for small teams, one for legacy system replacement, one for commercial off the shelf (COTS) system installations, and so on. This gives them a head start at tailoring the AUP to meet the exact needs of each project team because a lot of the common tailoring has already occurred.
Elaboration	Evolve the working environment. As the project progresses the understanding of the requirements evolves, the architectural strategy evolves, and the overall focus. The end result is that we will need to evolve the environment by installing new tools, or removing tools we no longer need.

	Tailor the process materials. We should tailor the AUP to meet the needs of the team. This may involve editing the AUP process materials (e.g. this page), we may choose to write a short document indicating what we won't be doing, or we may simply choose to "do the right thing at the right time".
Construction	Support the team. Project team members will need help using and/or configuring various tools to meet their needs. They will also need help to choose the proper documentation templates and follow the enterprise guidance. Evolve the working environment. See above. Setup training environments. As the deployment planning progresses we may discover that we need to train end users, support staff, and/or operations staff. This training effort may require training rooms and/or training versions of the system available, often during the Transition phase. We may need to start setting these environments up towards the end of the Construction phase.
Transition	Setup operations and/or support environments. Support staff, and sometimes operations staff, often need to have a version of the system setup which they use to simulate report defects in a safe manner. Recover software licenses. As the project comes to completion the team may need to uninstall software licenses from peoples workstations who no longer need the software so that the licenses may be available to others within the organization.

Milestones According to Methodology

As we can see in Figure 1, there are four milestones in the AUP. At each of these milestones, which signal the end of the phase, the Project should consider having a "milestone review" which verifies that the team has successfully fulfilled the milestone criteria. The four milestones are:

1. Lifecycle Objectives (LCO)
2. Lifecycle Architecture (LCA)
3. Initial Operating Capacity (IOC)
4. Product Release (PR)

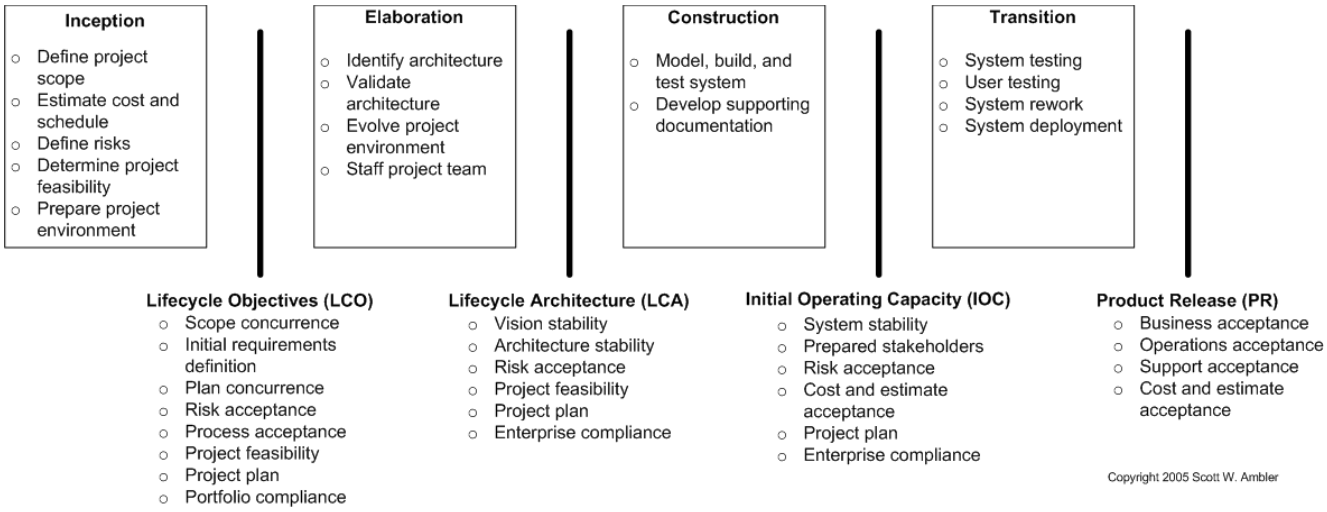


Figure 11 - The Phases and Milestones of the AUP

Inception Phase Milestone: Lifecycle Objectives (LCO)

At this milestone, the stakeholders assess the state of the project. They must agree on the following:

- **Scope concurrence**. The stakeholders reach agreement as to the scope of the project.
- **Initial requirements definition**. There is agreement that the right set of requirements have been captured, at a high level, and there is a shared understanding of those requirements.
- **Plan concurrence**. The stakeholders agree with the initial cost and schedule estimates.
- **Risk acceptance**. The risks have been identified, assessed, and acceptable strategies to address them have been identified.
- **Process acceptance**. The AUP has been initially tailored and agreed to by all parties.
- **Feasibility**. The project makes sense from business, technical, and operational perspectives.
- **Project plan**. Adequate plans are in place for the next phase (Elaboration).
- Portfolio compliance. Does the scope of the project fit well into the organization's overall project portfolio?

Realization Phase / Elaboration sub-Phase Milestone: Lifecycle Architecture (LCA)

At this milestone, the stakeholders assess the state of the project. They must agree on the following:

- **Vision stability**. The project vision has stabilized and is realistic.

- **Architecture stability**. We agree that the architecture is stable and sufficient to satisfy the requirements. The architecture has been prototyped where appropriate to address major architectural risks.
- **Risk acceptance**. The risks have been assessed to ensure they have been properly understood and documented and strategies to handle them are acceptable.
- **Feasibility**. The project still makes sense from business, technical, and operational perspectives.
- **Project plan**. Detailed iteration plans for the next few Construction iterations, as well as a high-level project plan, are in place.
- **Enterprise compliance**. Does the system architecture reflect the realities of the enterprise architecture?

Realization / Construction Phase Milestone: Initial Operational Capability (IOC)

At this milestone, the stakeholders must agree on the following:

- **System stability**. The software and supporting documentation are acceptable (stable and mature) to deploy the system to users.
- **Prepared stakeholders**. The stakeholders (and the business) are ready for the system to be deployed (although may still need training).
- **Risk acceptance**. The risks have been assessed to ensure they have been properly understood and documented and strategies to handle them are acceptable.
- **Cost and estimate acceptance**. The current expenditures are acceptable and reasonable estimates have been made for future costs and schedules.
- **Project plan**. Detailed iteration plans for the next few Transition iterations, as well as a high-level project plan, are in place.
- **Enterprise compliance**. Do the work products produced by the team comply to the appropriate enterprise standards?

Finalization/Transition Phase Milestone: Product Release (PR)

At this milestone, the stakeholders must agree on the following:

- **Business stakeholder acceptance**. The business stakeholders are satisfied with, and accept, the system.
- **Operations acceptance**. The people responsible for operating the system once it is in production are satisfied with the relevant procedures and documentation.
- **Support acceptance**. The people responsible for support the system once it is in production are satisfied with the relevant procedures and documentation.

- **Cost and estimate acceptance**. The current expenditures are acceptable and reasonable estimates have been made for future production costs.

Roles

In order to understand roles it is important to understand:

1. Roles can be held by multiple people.
2. A single person can take on multiple roles.
3. A role is not a position.
4. We should strive to become a generalizing specialist who has one or more specialties (e.g. database administration, project management, ...), a general knowledge of the overall software process, and a good understanding of the domain in which we work.
5. Not all roles will be presented in the Organizational Scheme. In the Scheme only key roles will be shown. Organizational Scheme in this document is more related to Project Management and Project Organization that to implementation methodology.

Role	Description	Discipline(s)
Agile DBA	A database administrator (DBA) who works collaboratively with project team members to design, test, evolve, and support the application's data schema(s).	Implementation
Agile Modeler	Someone who creates and evolves models, be they sketches, index cards, or complex CASE tool files, in an evolutionary and collaborative manner.	Model Implementation
Anyone	Any person in any other role.	Configuration Management Project Management
Configuration Manager	A configuration manager is responsible for providing the overall CM infrastructure and environment to the development team.	Configuration Management
Deployer	A deployer is responsible for deploying the system into pre-production and production environments.	Deployment
Developer	A develop writes, tests, and builds software.	Model

		Implementation Deployment
Process Engineer	Develops, tailors and supports the organizations software process materials (process descriptions, templates, guidance, examples, ...).	Environment
Project Manager	Manages the team members, protects the team members, builds relationships with stakeholders, coordinates interactions with stakeholders, plans, manages and allocates resources, shapes priorities, and keeps the team focused.	Model Test Deployment Project Management
Reviewer	Evaluates project work products, often "works in progress", providing feedback to the team.	Test
Stakeholder	A project stakeholder is anyone who is a direct user, indirect user, manager of users, senior manager, operations staff member, support (help desk) staff member, developers working on other systems that integrate or interact with the one under development, or maintenance professionals potentially affected by the development and/or deployment of a software project.	Model Implementation Test Deployment Project Management
Technical Writer	Technical writers are responsible for producing stakeholder documentation such as training materials, operations documentation, support documentation, and user documentation.	Deployment
Test Manager	Test managers are responsible for the success of the testing effort, including planning, management, and advocacy for testing and quality activities.	Test
Tester	Testers are responsible for writing, conducting, and logging the outcomes of testing efforts.	Test
Tool Specialist	Tools specialists are responsible for selecting, acquiring, configuring, and supporting tools.	Environment

Deliverables

The Agile UP distinguishes between:

- Deliverable which we absolutely must produce as a permanent system work product.
- Other project work products which we will likely discard because we do not want to maintain it over time.
- Enterprise work products which are maintained within the IT department and shared across projects.

Minimum Deliverables

The minimum deliverables for an Agile UP system are described, in priority order, in the following table:

Deliverable	Description
System	The working software, hardware, and documentation to be deployed into production.
Source Code	The program code for the system.
Regression Test Suite	A collection of test cases, and the code to run them in the appropriate order. The regression test suite will include a wide range of tests, including acceptance tests, unit tests, system tests, and so on.
Installation Scripts	Code for installing the system into the (pre-)production environment.
System Documentation	The documentation delivered as part of the system to help stakeholders work with it and developers to maintain and evolve it. Potentially comprised of operations, support, user, and system overview documentation.
Release Notes	The release notes should summarize the "good things to know" about the current release of the system which you're building.
Requirements Model	Describes the requirements which the system should fulfill. Comprised of a variety of work products, potentially including acceptance tests, automation opportunities, business process model, business rules, domain model, organization model, project glossary, technical requirements, use case model, and a user interface model.

Design Model	Describes the design of the system. Comprised of a variety of work products, potentially including a deployment model, an object model, a physical data model (PDM), a security threat model, a system overview document, and a user interface model.

Other Project Work Products

Other work products, listed in alphabetical order, which may be chosen to create or use as a matter of course include:

Work Product	Description
Acceptance Tests	Acceptance tests describe black-box requirements, identified by the project stakeholders, which the system must conform to.
AUP	The Agile Unified Process.
Automation Opportunities	An indication of manual activities which could potentially be automated.
Budget	An indication of the amount of funds, when we will receive the funds, and the criteria (if any) the team must meet to receive the funds, to support the project efforts.
Business Process Model	A description of the business activities, the information flows between them, and the sources/destinations of the information.
Business Rules Specification	A business rule specification captures the collection of business rules implemented by a system. A business rule defines or constrains one aspect of the business that is intended to assert business structure or influence the behavior of the business. Business rules often focus on access control issues, business calculations, or policies of the organization.
Data Schema	The schema of the data sources. In the case of relational databases, it's described by data definition language (DDL), for XML data sources you'll use XML schema definitions or XML DTDs.

Defect Report	A type of change request that defines a system-related problem; that is, the system is not working the way that it is supposed to.
Deployment Model	Describes how we will organize the hardware, middleware, and software aspects of the system.
Deployment Plan	Describes the approach to deploying the system into production.
Domain Model	Depicts the main business entities, the relationships between them, and potentially their responsibilities.
Estimate	The projected cost for completing the project.
Individual's Estimate	An estimate made by an individual of the time and/or cost to perform a task.
Object Model	Describes the object schema, the code which comprises the software. Often comprised of several views for the static structure and the dynamic aspects of the schema.
Operations Documentation	Captures the procedures and supporting information to operate the system once it is in production.
Organization Assessment	Describes the organization (perhaps a division of the company) into which the system will be deployed, including an indication of the organization's ability to adopt and use the new system.
Organization Model	Indicates the people involved with the project and the reporting structure between them. Should indicate both the members of the project team as well as the major stakeholders and their roles.
Physical Data Model (PDM)	Describes the physical schema of a data store, such as a relational database or XML file.
Project Glossary	Describes the critical business and technical terms on the project.

Project Overview	Summarizes the goals, plan, and mission of the project. Potentially comprised of a project goals statement, project vision statement, and an organization assessment.
Project Plan	Comprised of iteration plans, the overall project schedule, risk list, estimate, and budget.
Project Resources	Comprised of funding, hardware/software, and facilities (such as rooms).
Project Schedule	Indicates the activities, the dependencies between them, and the project milestones.
Proof-of-Concept Prototype	Working code that shows that proves that a technical approach works.
Review Record	The results, including assigned action items, of a review.
Risk List	A list of identified risks, and mitigation strategies (if any).
Security Threat Model	A model which explores security threats to the system.
Support Documentation	The documentation required by the support staff, such as trouble shooting guides, contact information for the development team, which enables them to support end users.
System Overview Document	Technical documentation for the people responsible for maintaining and evolving the system.
Tailored Process	Comprised of the AUP, guidance, and templates tailored to the needs of the project team.
Technical Requirements	Describes non-behavioral issues such as usability, security, and performance. Often referred to as "non-functional" requirements.
Templates	An "electronic form" containing common information fields to fill out for a given kind of work product.
Test Model	Comprised of a Regression Test Suite and any Defect Reports.
Test Strategy	A description of the general approach to testing.

Tools	The software packages used to develop the system.
Training Materials	The course notes, overview documentation, assignments, and so on used to help train end users, support staff, and operations staff to work with the system.
Use Case	Use cases describe something of value to a stakeholder and are a primary requirements work product of the Agile UP.
Use Case Model	A use case model is comprised of zero or more use case diagrams, use case descriptions, and actor descriptions.
User Documentation	The manuals, help documentation, and so on that end users use to help them understand the system.
User Interface Model	Describes the user interface of the system.

Application Development Approach

Overview

As the name implies, AMDD is the agile version of Model Driven Development (MDD). MDD is an approach to software development where extensive models are created before source code is written. A primary example of MDD is the Object Management Group (OMG)'s Model Driven Architecture (MDA) standard. With MDD a serial approach to development is often taken, MDD is quite popular with traditionalists, although as the RUP/EUP shows it is possible to take an iterative approach with MDD. The difference with AMDD is that instead of creating extensive models before writing source code we instead create agile models which are just barely good enough that drive the overall development efforts. AMDD is a critical strategy for scaling agile software development beyond the small, co-located team approach that we saw during the first stage of agile adoption.

Figure 12 - The AMDD lifecycle: Modeling activities throughout the lifecycle of a projectdepicts a high-level lifecycle for AMDD for the release of a system. First, let's start with how to read the diagram. Each box represents a development activity. The envisioning includes two main sub-activities, initial requirements envisioning and initial architecture envisioning. These are done during iteration 0, iteration being another term for cycle or sprint. "Iteration 0" is a common term for the first iteration before we start into development iterations, which are iterations one and beyond (for that release). The other activities – iteration modeling, model storming, reviews, and implementation – potentially occur during any iteration, including iteration 0. The time indicated in each box represents the length of an average session: perhaps you'll model for a few minutes then code for several hours.

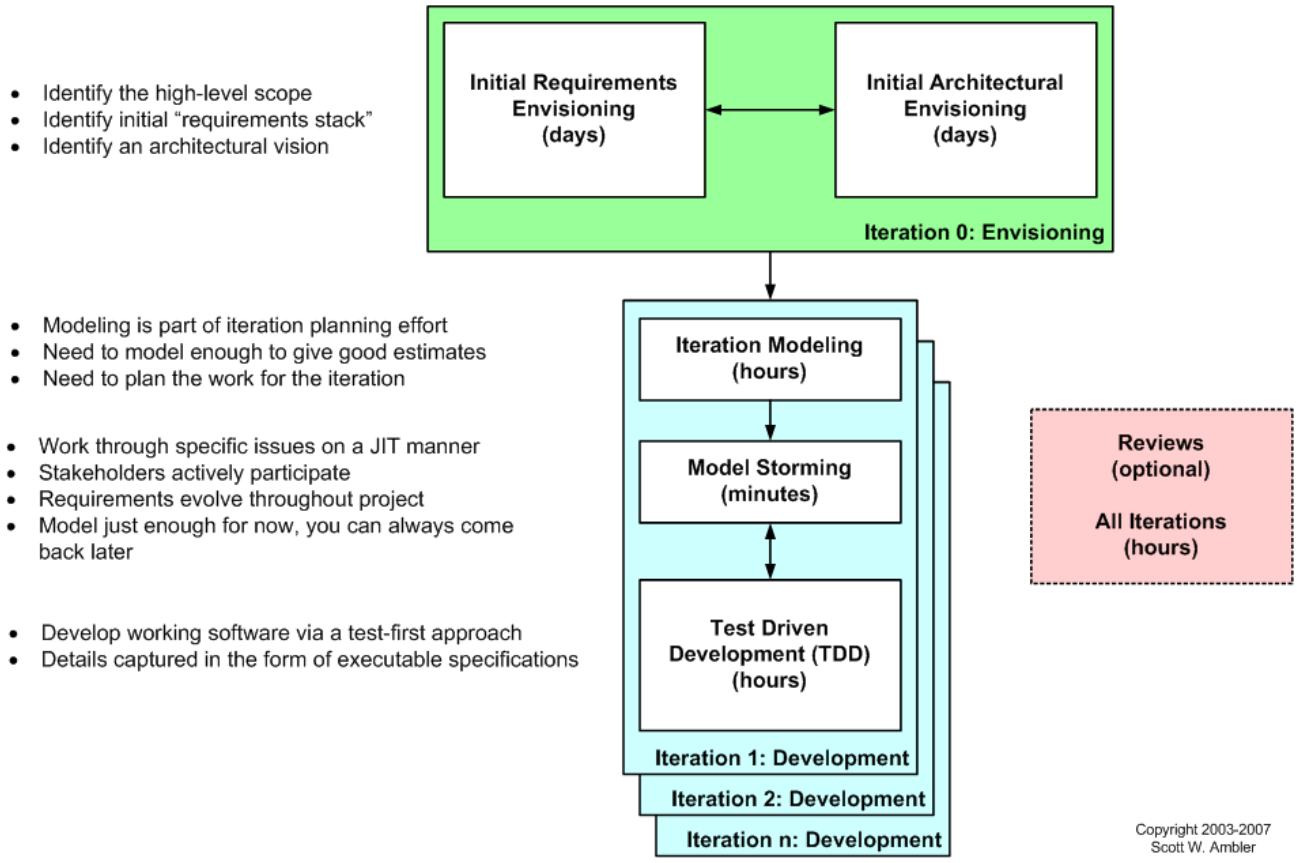


Figure 12 - The AMDD lifecycle: Modeling activities throughout the lifecycle of a project

Figure 13 depicts how the AMDD activities fit into the various iterations of the agile software development lifecycle. It's simply another way to show that an agile project begins with some initial modeling and that modeling still occurs in each construction iteration.

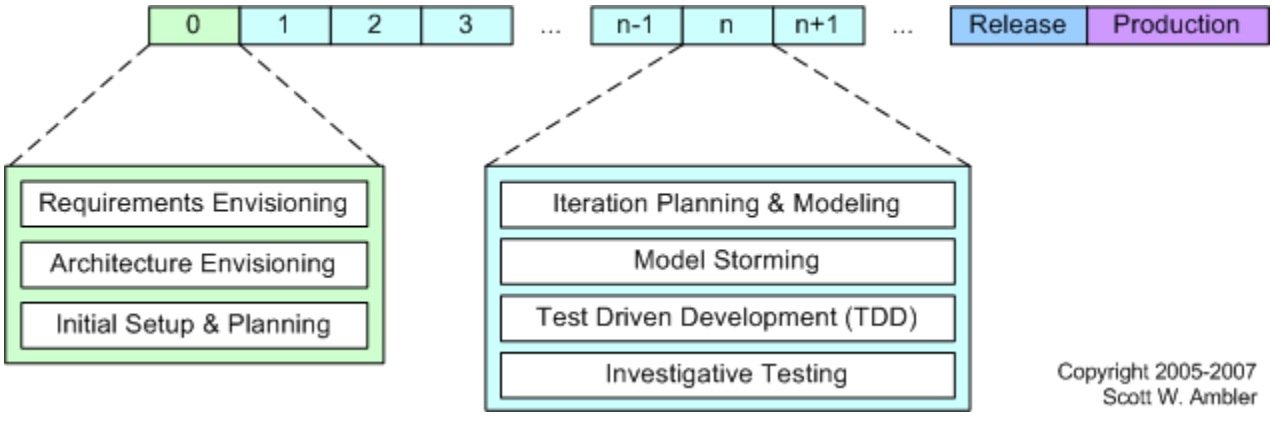


Figure 13 - AMDD Through the Agile Development Lifecycle

Envisioning

The envisioning effort is typically performed during the first week of a project, the goal of which is to identify the scope of the system and a likely architecture for addressing it. To do this we will do both high-level requirements modeling and high-level architecture modeling. The goal isn't to write detailed specifications, that proves incredibly risky in practice, but instead to explore the requirements and come to an overall strategy for the project. For short projects (perhaps several weeks in length) we may do this work in the first few hours and for long projects (perhaps on the order of twelve or more months) we may decide to invest two weeks in this effort. We highly suggest not investing any more time than this as we run the danger of over modeling and of modeling something that contains too many problems (two weeks without the concrete feedback that implementation provides is a long time to go at risk, in my opinion).

Initial Requirements Modelling

For the first release of a system we need to take several days to identify some high-level requirements as well as the scope of the release (what we think the system should do). The goal is to get a good gut feel what the project is all about. For the initial requirements model my experience is that we need some form of usage model to explore how users will work with the system, an initial domain model which identifies fundamental business entity types and the relationships between then, and an initial user interface model which explores UI and usability issues.

The goal is to build a shared understanding, it isn’t to write detailed documentation. A critical success factor is to use inclusive modeling techniques which enable active stakeholder participation.

Initial Architecture Modelling

The goal of the initial architecture modeling effort is to try to identify an architecture that has a good chance of working. This enables we to set a (hopefully) viable technical direction for the project and to provide sufficient information to organize the team around the architecture (something that is particularly important at scale with large or distributed teams).

On the architecture side of things often creation of free-form diagrams which explore the technical infrastructure is usual and initial domain models to explore the major business entities and their relationships, and optionally change cases to explore potential architecture-level requirements which the system may need to support one day. In later iterations both the initial requirements and the initial architect models will need to evolve as Project learns more, but for now the goal is to get something that is just barely good enough so that the team can get going. In subsequent releases it may decided to shorten iteration 0 to several days, several hours, or even remove it completely as the situation

dictates. The secret is to keep things simple. The Model will not have a lot of detail. In the writing of use cases this will mean that point-form notes are good enough. At domain modeling in sketch or collection of CRC cards is likely good enough. For the architecture a whiteboard sketch overviewing how the system will be built end-to-end is good enough.

Many traditional developers will struggle with an agile approach to initial modeling because for years they've been told they need to define comprehensive models early in a project. Agile software development isn't serial, it's iterative and incremental (evolutionary). With an evolutionary approach detailed modeling is done just in time (JIT) during development iterations in model storming sessions.

Iteration Modelling

At the beginning of each Construction iteration the team must plan the work that they will do that iteration. An often neglected aspect of Mike Cohn's planning poker is the required modeling activities implied by the technique. Agile teams implement requirements in priority order, see Figure 14, pulling an iteration's worth of work off the top of the stack. To do this successfully we must be able to accurately estimate the work required for each requirement, then based on the previous iteration's velocity (a measure of how much work we accomplished) the Team pick that much work off the stack. For example, if last iteration accomplished 15 points worth of work then the assumption is that all things being equal we'll be able to accomplish that much work this iteration. This activity is often referred to as the "planning game" or simply iteration planning.

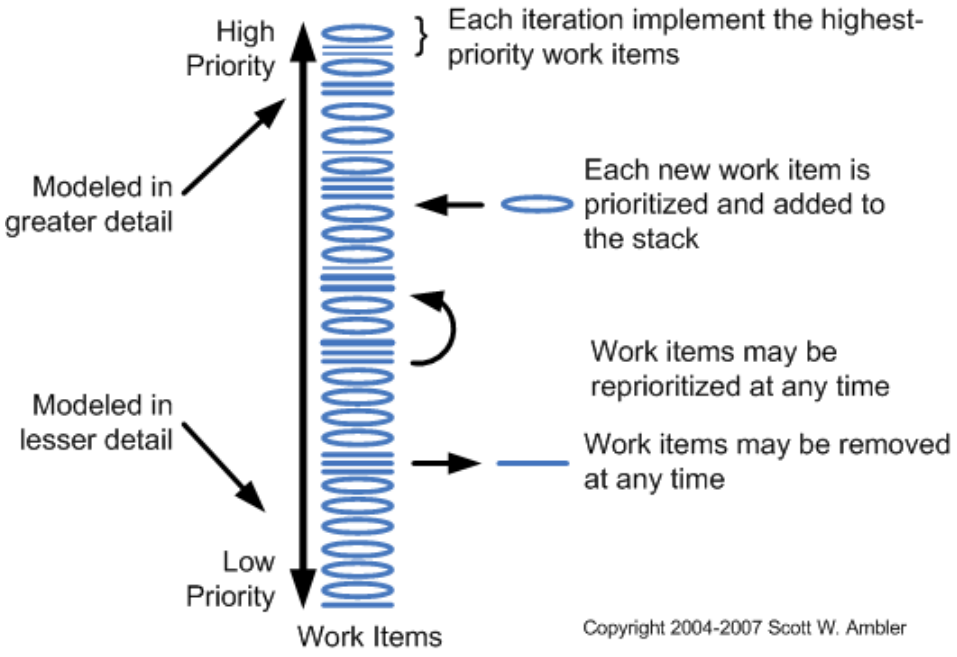


Figure 14 - Agile requirements change management process

To estimate each requirement accurately the Team must understand the work required to implement it, and this is where modeling comes in. This modeling in effect is the analysis and design of the requirements being implemented that iteration.

Model Storming: Just In Time (JIT) Modeling

The vast majority of modeling sessions involve a few people, usually just two or three, who discuss an issue while sketching on paper or a whiteboard. These "model storming sessions" are typically impromptu events, one project team member will ask another to model with them, typically lasting for five to ten minutes (it's rare to model storm for more than thirty minutes). The people get together, gather around a shared modeling tool (e.g. the whiteboard), explore the issue until they're satisfied that they understand it, then they continue on (often coding). Model storming is just in time (JIT) modeling: identify an issue which needs to be resolved, quickly involvement of few team mates who can help, the group explores the issue, and then everyone continues on as before. Extreme programmers (XPers) would call modeling storming sessions stand-up design sessions or customer Q&A sessions.

Executable Specification via Test Driven Development (TDD)

During development it is quite common to model storm for several minutes and then code, following common Agile practices such as Test-First Design (TFD) and refactoring, for several hours and even several days at a time to implement what you've just modeled. For the sake of discussion test-driven design (TDD) is the combination of TFD and refactoring. This is where the team will spend the majority of its time, something that Figure 1 unfortunately doesn't communicate well. Agile teams do the majority of their detailed modeling in the form of executable specifications, often customer tests or development tests. Why does this work? Because the model storming efforts enable the Team to think through larger, cross-entity issues whereas with TDD we think through very focused issues typically pertinent to a single entity at a time. With refactoring we evolve the design via small steps to ensure that the work remains of high quality.

TDD promotes confirmatory testing of the application code and detailed specification of that code. Customer tests, also called agile acceptance tests, can be thought of as a form of detailed requirements and developer tests as detailed design. Having tests do "double duty" like this is a perfect example of single sourcing information, a practice which enables developers to travel light and reduce overall documentation. However, detailed specification is only part of the overall picture – high-level specification is also critical to the success, when it's done effectively. This is why we need to go beyond TDD to consider AMDD.

The Project may even require to "visually program" using a sophisticated modeling tool such as Rational Software Architect (RSA).

Why Does This Work?

1. **We can still meet the "project planning needs"**. By identifying the high-level requirements early, and by identifying a potential architecture early, we have enough information to produce an initial cost estimate and schedule.
2. **We manage technical risk**. The initial architecture modeling efforts enable we to identify the major areas of technical risk early in the project without taking on the risk of over modeling the system. It's a practical "middle of the road" approach to architectural modeling.
3. **We minimize wastage**. A JIT approach to modeling enables we to focus on just the aspects of the system that you're actually going to build. With a serial approach, we often model aspects of the system which nobody actually wants.
4. **We ask better questions**. The longer we wait to model storm a requirement, the more knowledge you'll have regarding the domain and therefore you'll be able to ask more intelligent questions.
5. **Stakeholders give better answers**. Similarly, the stakeholders will have a better understanding of the system that you're building because you'll have delivered working software on a regular basis and thereby provided them with concrete feedback.

Approaches to AMDD

There are three basic approaches to applying AMDD on a project:

1. Manual. Simple tools, such as whiteboards and paper, and inclusive models are used for modeling. This is likely 70-80% of all business application modeling efforts.
2. Design Tool. Inclusive models are used to explore requirements with stakeholders, and to analyze those requirements. Developers then use sophisticated modeling tool for detailed design, (re)generating source code from the models. This is likely 20-30% of all business application modeling efforts.
3. Agile MDA. Very sophisticated, MDA-based modeling tools used to create extensive models from which working software is generated. At best this approach will be used for 5-10% of business application modeling efforts.

Acceptance Testing Plan

Testing and testing methodology

Testing

Testing of the system will be divided into several testing levels:

- *Unit testing* – will demonstrate the quality of each individual unit. From the beginning, each business logic component developed will be tested continuously using unit testing tools (such as CSUnit) for the compliance with the Requirements specification. In unit testing the developers use so called “black box testing”, i.e. they fill in input data, observe output data, but do not know, or pretend not to know, how the program works. The testing team looks for uncommon input data and conditions that might lead to uncommon outputs. Input data are “interesting” representatives of a class of possible inputs if they are the ones most likely to make visible an error in the program. The unit testing is part of the development process – writing code and debugging is an iterative cycle of programming.
- *Component Testing* – is performed by the testing team after unit testing. In this test the program is examined part by part (i.e. component by component). The component testing is the first cycle of the testing process. It inspects a particular component for proper functionality and proper working. For each component the testing team executes a specific test scenario to check how the component reacts to typical and uncommon inputs or environment events.
- *Integration testing* is carried out to demonstrate that the functionality of a subsystem, including communication and security, is fulfilling the requirements. From the early stages of the project, testing infrastructure will be setup for the possibility of integration testing by developers and for the ongoing possibility of Beneficiary to see the application development progress. The integration testing is a more thorough release test. It provides an opportunity to review the solution functionality in details before the solution deployment. The testing team carefully compares the program, the project documentation, and the Requirements specification. The Integration testing can be performed immediately after the component testing to inspect how the components work together. The testing team should ensure proper functionality, including navigation, data entry, processing and retrieval for all related components.
- *Regression testing* is used in two different ways. Common for both is the idea of reusing old tests. (1) An error is found, fixed and then the test that made the issue visible in the first place is repeated. Under these circumstances, the regression testing is done to make sure that the fixed code works properly. (2) An error is found, fixed and tested and then a standard series of tests is executed to make sure that the change didn’t disturb anything else. Both types of the regression tests are executed whenever errors are fixed.
- *System testing* follows the development: specialized testers (not from the pool of developers) examine the complete behavior of the system, using the Quality plan written in advance.

- *User interface testing* is a specific testing routine. When testing the UI the testing team measures the following items:
    - Functionality – does the UI fit the clients requirements;
    - UI reaction time – is the UI reaction time in admissible bounds;
    - Availability – does the UI work adequately in any possible cases;
    - Usability – the UI should have intuitive navigation and no extra steps;
    - Look & feel – does the UI match project design standards.
- *Performance testing* – will confirm the operational condition and performance requirements of the system. The test cases and test plans specified for this test phase will be reused during user acceptance test. The Performance tests identify tasks and measure how long it takes to do each. One objective of performance testing is performance enhancement. These tests can determine which modules are executed most often or use the largest part of the computer resources (time and storage). These modules should be examined in detail and if necessary, rewritten to run more quickly. The solution's bad performance may reflect bugs, especially when there are client's expectations for high performance of some modules or components.
- *User acceptance testing* – carried out per subsystem - will demonstrate the full scope of the proposed system and confirm compliance to the requirements as detailed in the Requirements and Technical specifications. From the pilot phase of the project end users will continuously provide feedback on system bugs and enhancements.
- *Solution Documentation review* can include: User Documentation, Technical Documentation, Design Documents. The testing team shall review and verify all documents, related to the project.
- *Automated testing* - most tests can be executed using specific test tools. For particular project or test types the testing team will use different tools to automate the testing process. The automated testing consists of writing specific commands, scripts, which should be executed to simulate a specific test – system, load, performance, etc. Test scripts will be prepared and used for regression testing. The automated testing saves much time for manual testing, reduces the risk in software development and reduces mistakes made by the testing team during the testing process.

Testing methodology

Quality Assurance System

Exact testing procedures / methods will be specified during the Design phase in the Quality plan.

In this section is briefly described the testing approach that is carried out by the Tenderer in large-scale integration projects, and which is envisaged for this Project. The specific testing approach for this Project, and more details will be specified in the Quality plan, which is a deliverable item and has to be accepted by the Beneficiary. For each subsystem of the Project corresponding test plans will be specified and reviewed (see the chart below). Tests will be carried out according to these test plans.

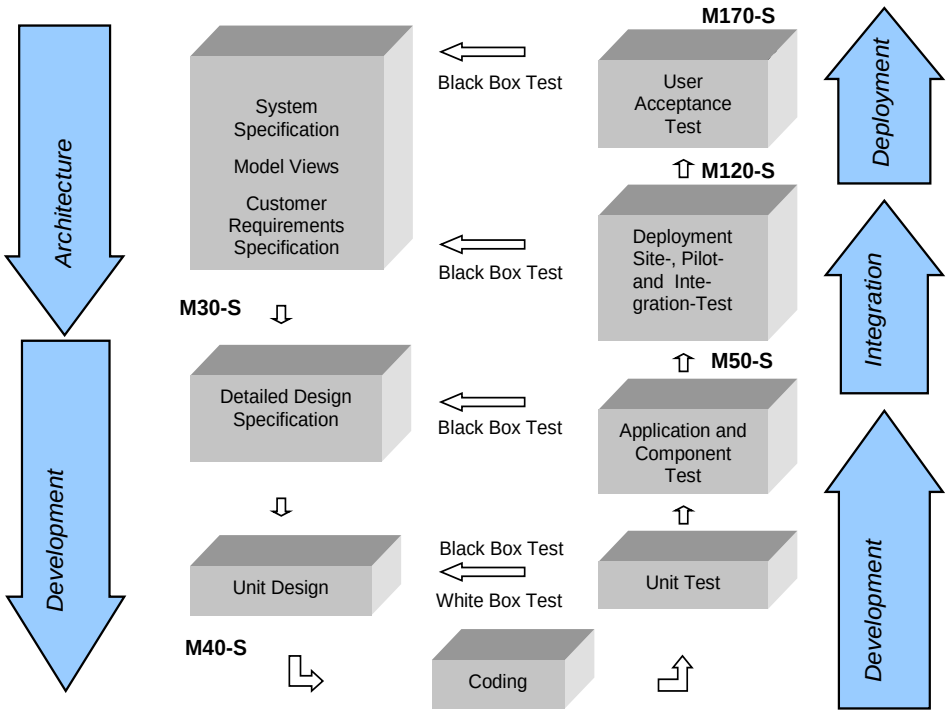


Figure 15 – Quality Assurance graph

The test methods used are identical in every subsystem and are briefly specified in the following chapter. For each subsystem a test suite containing the test cases is developed. This test suite will be enlarged and completed during the test process and can be reused during user acceptance testing and regression testing after bug fixing.

Test Responsibilities

The development department together with the testing team is responsible for unit testing and application component testing (operational testing) until end of development. The Tenderer's testing team is responsible for subsystem testing (pilot tests) and deployment site testing. These tests will be carried out during pilot deployment phase.

Test Methods

Two kinds of methods are used to define test cases:

- White Box Test – the white box test implies knowledge of the structure of the test object. The detailed Technical specification and the source code of the test object are used to define test cases. This form of testing will be used in addition to black box testing at component level and exclusively at unit level;
- Black Box Test – for a black box test the tester does not use any information about the structure of the test object or how it has been developed. The appropriate specification – customer requirement specification, detailed functional specification – is used to define all the test cases.

Test Tools

The specific test tools that will be used during testing will be defined in the Quality plan.

Test Process for a subsystem

At each design level test plans will be written to specify tests appropriate to the level of design. The Hardware and Operating Systems conformance to the Project requirements will be verified during the installation process.

At the highest level, there is the Technical specification and the Beneficiary's Requirements specification for the Project and a user acceptance test is specified to verify that the system meets those agreed requirements.

Lower down a deployment site test is defined, reviewed and agreed upon to verify that the system conforms to the model views. Tests down to parts of the application test level are designed, developed and carried out by a test team, including representatives of the Beneficiary.

This process is carried out down to the function and module level, where a unit test is specified (white box test by development team).

The testing is then carried out in reverse order, from the bottom up. Once a group of units has successfully completed its tests, it is integrated into components for component testing. Larger and larger groups of validated software components are progressively integrated for testing at higher levels, until the complete subsystem has been validated, and the agreed acceptance test is carried out.

Error Handling

Error handling is based on an iterative procedure, described below.

Errors, found by the testing team, are reported to the development team. The following items are obligatory for the error description:

- Area, screen, page or component in which error appears;
- Issue category – for example: error/bug, improvement, enhancement, change, etc.;
- Error priority – for example: ASAP, high, medium, low;
- Issue state – for example: identified, in progress, finished, verified, closed, etc.;
- Solution build number, version or release;
- Reported by person – name and project role;
- Assigned to person – name and project role;

The following items are optional and recommendable for the testing team:

- Attachments – doc files, images, etc.;
- Issue/error severity – for example: critical, major, normal, minor, etc.;
- Suggested fix;
- Opinion or idea when can the mistake be found and how to debug it;
- Additional comments.

After the error is corrected by the development team, the respective system unit is submitted back to the testing team for regression testing.

Such iterations are performed until all defects are fixed.

Testing Criteria

The following testing criteria will be applied when developing test plans, test specifications, (automatic) test software and in performing the tests:

- The basis for the definition of test goals and test plans are the documents created during Design and Development phase;
- Repeatability:
    - every test case will be reproducible;
    - the expected test result for every test case will be defined.
- Test execution and test results:
    - the execution of tests will be recorded;
    - success and failure of tests will be logged;

- where applicable, problem reports will be raised and traced to the test cases;
- summaries of test executions will be created.

Quality Criteria

The following quality criteria will be applied when providing an overall assessment for the software. Specific indicators for each criterion will be defined in the Quality plan.

- Correctness / reliability - the degree to which the software meets the documented requirements and specifications;
- Efficiency - how effective is the software's use of system resources and what is the execution speed;
- Usability - the ease with which a user can learn to operate the software and interpret its results;
- Maintainability - how easy is it to make changes to the documentation and software in order to keep it up to date or to locate and fix an error;
- Testability - how easy is it to test the software;
- Portability - how easy is it to move the software to a new hardware / software environment;
- Reusability - how easy is it to use all of the software or parts of it in other applications;
- Integrity - how secure is the software against attempts to breach its access controls;
- Interoperability - how easy is it to connect the software to other systems to exchange data with them.

Test Cases

Test cases in the form of test inputs, execution conditions, and expected results will be developed to reach particular objectives (e.g. to exercise a particular program path or to verify the compliance with a specific requirement).

The purpose of a test case will be to identify, define and communicate conditions that will be implemented in the test. Test cases are necessary to verify successful and acceptable implementation of the solution requirements (use cases).

A three-step process for creating test cases from a fully-detailed use case is described below:

- For each use case a full set of test scenarios will be created – the testing team will review the use case description in details and identify each combination of main and alternate flows (the test scenarios) and create a scenario matrix.
- For each scenario at least one test case and the conditions that will make it "execute" will be identified – the testing team will identify test cases by analyzing the test scenarios. For some scenarios a few cases can exist; for others – only one. The next step in fleshing out the test cases is to read the use case description and find the conditions or data elements required to execute the various scenarios.

- For each test case the data values to be used will be identified – the test cases cannot be implemented or executed without test data. Most important for test data identification and test cases design is to use the following techniques:
    - Equivalence class analysis;
    - Boundary analysis;
    - Testing state transitions;
    - Testing race conditions and other time equivalence testing;
    - Doing error guessing.

  The testing team does not need to add all possible values to find data errors during testing. The first task is to analyze data types and to define their boundaries. If the testing team uses boundary values this fact can spread a lot of error cases. Data entry will be analyzed with valid and invalid equivalence classes, boundary and special cases.

  The state transitions can be much more complex than menu-to-menu. For testing interactions between paths the testing team should select paths through the program as follows:

    - Test all paths that the actors are particularly likely to follow;
    - If the testing team has any reason to suspect that choices at one menu level or data entry screen may affect the presentation of choices elsewhere, the testing team can test the effects of those choices or entries;
    - Along with conducting the most urgent types of tests, as described above, the testing team can try a few random paths through the solution. The testing team selects different paths in each test cycle randomly.

  The QA team can try to guess all possible errors to inspect handling, for correct error messages and their presence. The QA team should include the expected outputs after errors appear in the test cases.

General Entry Criteria

The overall criteria to start in a test level are:

- All necessary documents must be available for the testers at a reasonably early time;
- Where applicable, testing specifications for this level must be defined and formally approved;
- Test resources (hardware, staff, etc.) must be available;
- Source code and where applicable make-files, configuration files or installation scripts must be available;
- System Parts that are used directly or indirectly by the test object and also its sub-objects must have passed the lower test levels successfully. The related test reports will be informally reviewed for acceptance.

More details of entry criteria will be specified in the respective test plans.

General Exit Criteria

The overall criteria required to successfully pass each test level are:

- All specified test cases have to be performed and the tests results must correspond to the expected values;
- Problems with lower priority which do not affect the functionality may remain outstanding while the test is classed as passed;
- Test reports must be completed;
- Software corrections must be regression tested.

More details of exit criteria will be specified in the respective test plans.

Test and Development Environment

For the testing purposes of the system, special environment will be setup, where new versions of components will be released. The test environment will consist of a complete set of testing tools, including software and hardware. This environment will also be used for regression tests and standard software patches deployment tests. This means that the application or any fixes will be installed on the productive system only after all tests are successfully passed on the testing system. The testing environment provided by Tenderer will be kept equal with the productive environment of the Beneficiary in the maximum possible extent.

Project Management

Management Approach

Project Management

Project management and controlling will be performed on the basis of the PMI methods framework. This is the tool used worldwide to manage and execute projects.

Since PMI can be applied to various sectors, the documentation alone is comprised of over 20 very comprehensive handbooks describing the individual phases and activities of project management. This process model is supported by various tools and document templates, including plans, reports, presentations, etc.

In summary PMI can be characterized as follows:

- **Business process driven approach:**
- Looking at the organization from the point of view of its business processes.
- **Communication and partnership:**
- Collaboration of consultants, IT staff and the technical field.
- **Changes are normal:**
- Our method assumes that the business environment and the user requirements undergo constant change.
- **A sequence of small successes is strived for:**
- The overall project success is the sum of many individual successes.

Following a brief introduction to the fundamental project split according to PMI, the three phases of running projects: project initiation, project execution and project completion, will be then dealt with in more detail.

Breakdown of Project Management and Controlling

PMI breaks down a project and thus the corresponding management activities into three consecutive phases:

- **Initiation;**
- **Execution, and**

- **Completion.**

Each of these phases and the activities contained therein will be subsequently briefly dealt with, whereby this offer can only provide a short overview of the individual activities.

The PMI documentation on project management is comprised of a total of several hundreds of pages and a corresponding deepening would go beyond the scope of this offer.

This scope results not only from the PMI claim to be a framework applicable to all types of project management and project controlling. It also means that not all activities principally foreseen in PMI have to be performed in every project. Rather, more of a tailoring of processes takes places during project initiation (see below) to make them fit the specific project by leaving out those steps which are not relevant or not essentially necessary for the implementation of the project

Project Initiation

The following illustration provides an overview of the central phases and activities of project initiation:

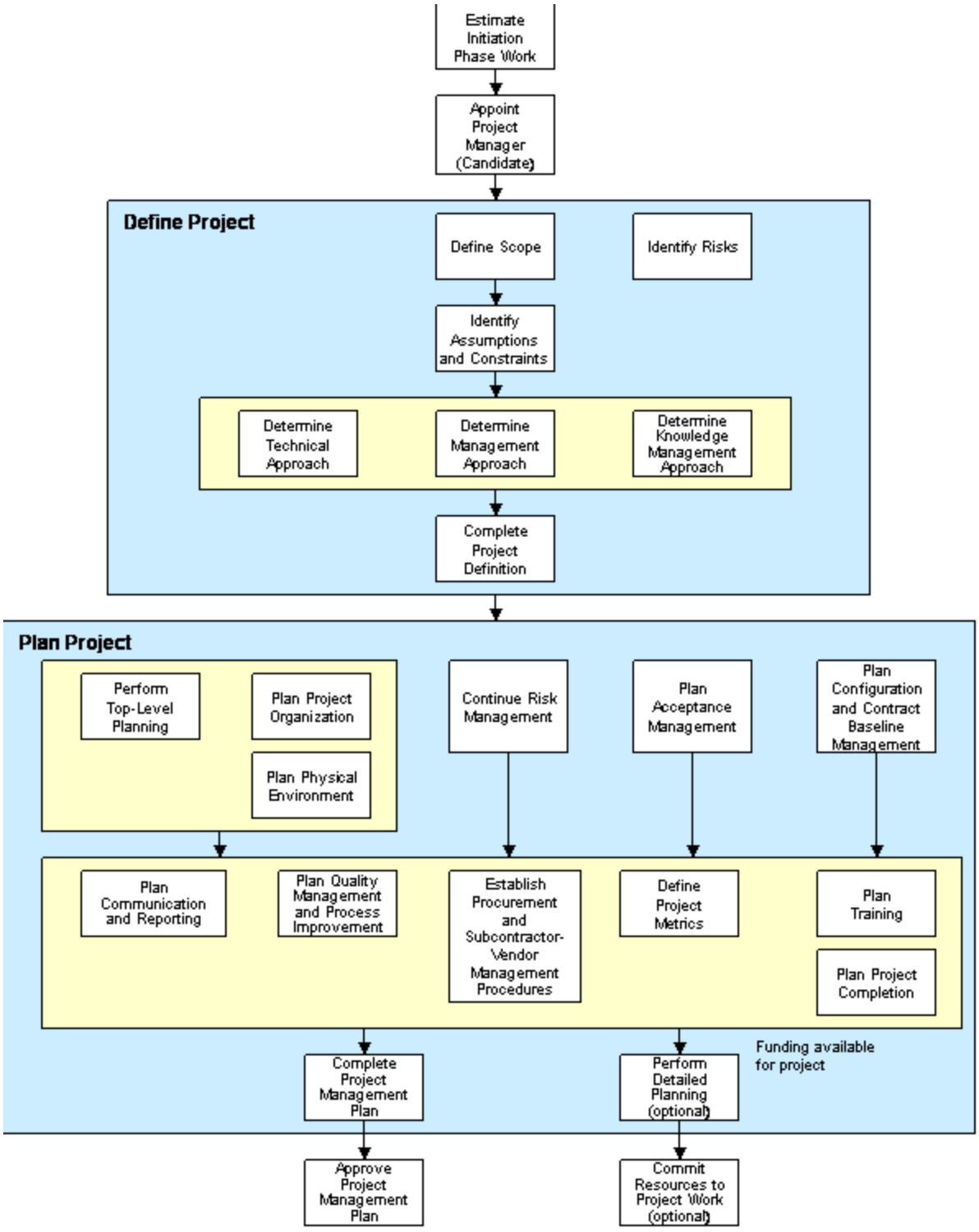


Figure 16 - Central phases and activities of project initiation

As shown on the illustration above, project initiation does not first start with the project start but rather significantly earlier. The project definition phase for this project has been already performed to a great degree in the course of drawing up the offer. The Proposal manager, respectively later the Project director is the person in charge of this.

The results of the project definition (i.e. the defined scope of project, the procedure for implementation from the technical and managerial point of view, the risks as well as the possibilities of knowledge transfer to the participating key persons) are manifested in the subsequent planning phase.

This starts with a project management plan at the top level and describes the project structure with its individual work packages and results, the foreseen resources and their breakdown as well as the scheduled periods and sequences for the individual activities.

The individual detailed plans covering the various project aspects, such as acceptance, training, quality management, communication or risks, are derived from this project management plan, whereby the planning for the near future (i.e. next one to two months) is always more precise than for activities which are further away (rolling planning).

The result of this planning phase is on one hand a project management plan comprising the significant objectives, tasks, responsibilities and measures from the point of view of the project management, as well as one or several detailed plans (e.g. for development, tests and quality assurance) which define the precise procedure during project realization based on the tender materials and our proposal, and which are defined and agreed in cooperation with the client at the begin of the project in the course of detailed planning.

On the basis of this precise procedure the allocation of the foreseen project resources to the individual areas of responsibility is then finalized.

Project Execution

The following illustration provides an overview of the central phases and activities of project execution:

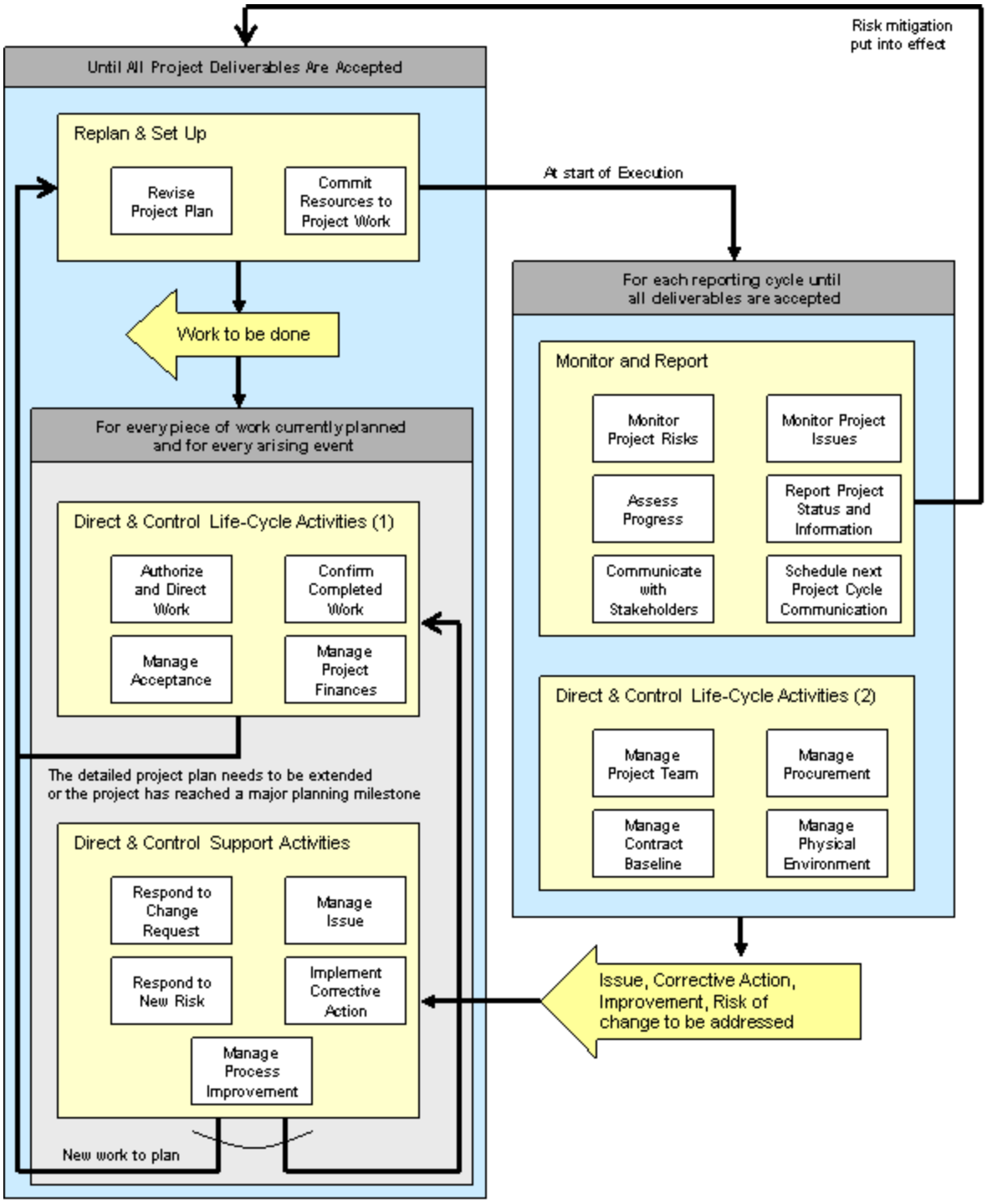


Figure 17 - Central phases and activities of project execution

In correspondence with the illustration above we understand project management and controlling during the actual project implementation, i.e. from the creation of the detailed specifications up to the final acceptance, as a sequence of cyclical activities which can be broken down into the following five areas:

- **adaptation of plans and resource allocation (planning cycle)**
- **control, tracing and completion of individual project tasks (operative cycle)**
- **reacting to unforeseen events (management cycle)**
- **monitoring of project progress and communication with the client and other stakeholders (control cycle)**
- **leading and managing staff and resources (leadership cycle)**

In detail this means that the project director (who in cooperation with the technical project manager and the training project manager) is responsible for the regular editing, completion and adaptation of the overall project plan and individual detailed plans to the actual project situation; and on the basis of the ongoing comparison between the target and actual data adapts targeted deployment of appropriate resources to the individual tasks and activities. These activities will be performed cyclically over the entire project again and again, until the project has been finally accepted.

On the "operative level" the tasks to be performed are steered and controlled result-oriented, rendered services and their results checked and then reviews and acceptance of individual deliverables packages (specifications, designs, software) held with the client. This also includes regular monitoring of deployed monetary means which has an effect on the planning and resources deployment in the same way as the project results.

In the course of a project, from time to time events occur which were unknown in the beginning and/or could not be foreseen, respectively planned. This includes for example additional requirements or modification of the existing ones, new risks which first appeared after project start, open issues which must be clarified and plan deviations due to processes not running optimally or missing resources. The project manager reacts to this by on one hand making decisions and taking or causing measures to get control over the situation, as well as by adjusting on the other hand the planning and resources allocation.

The results of planning, realization and the decisions are regularly verified with regard to their impacts on the plans respectively their topicality and status, and documented in corresponding project reports. These are not only a tool of project controlling but rather a means to communicate with external (as understood by the client) stakeholders to whom the client regularly reports.

Finally, the project director is also responsible for leading and motivating the project staff they are in charge of, as well as for compliance with the contractual project basis and procurement of necessary supplies and services in good time.

Project Completion

The following illustration provides an overview of the central phases and activities of project completion:

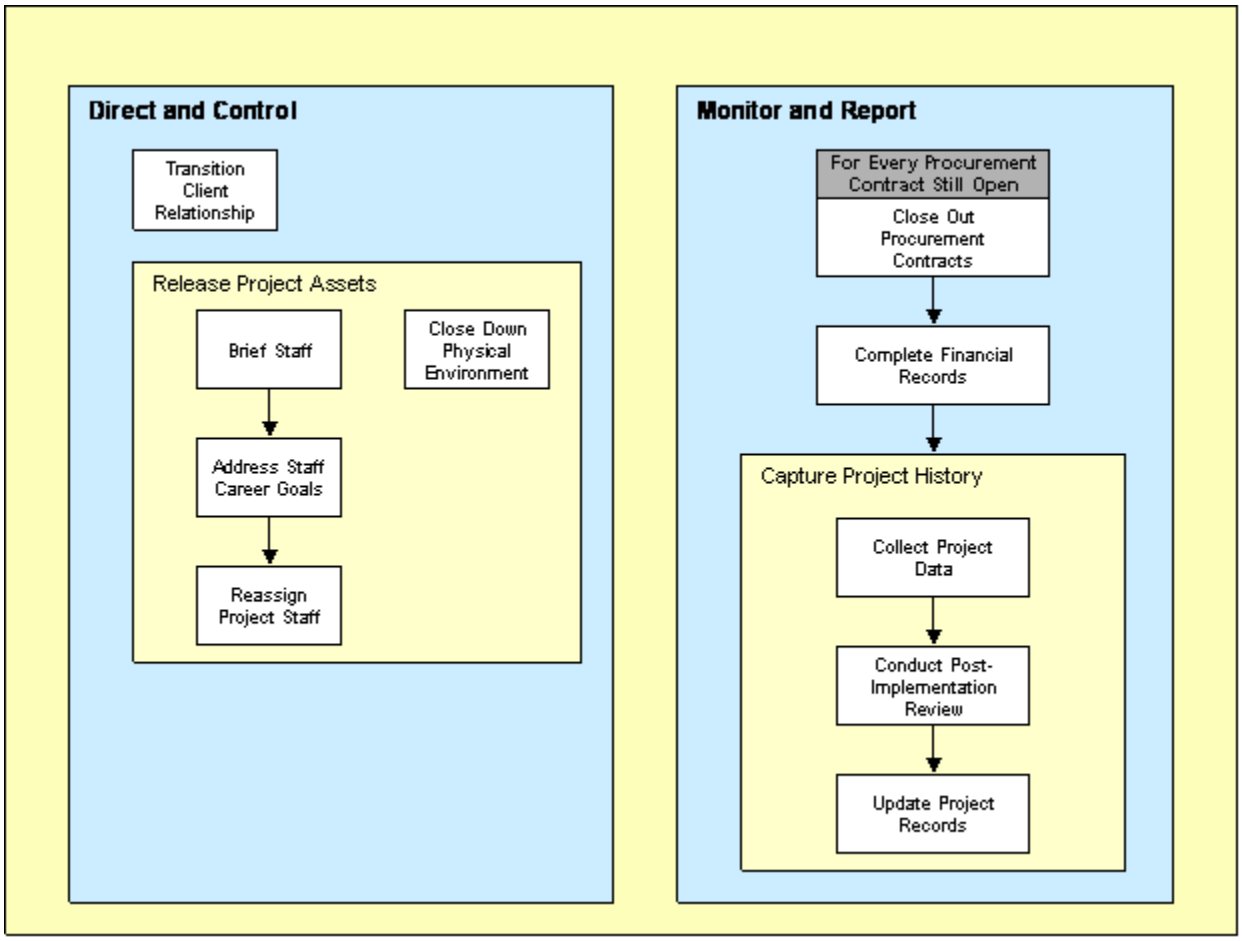


Figure 18 - Central phases and activities of project completion

The project completion phase starts after the project has been finally accepted by the client and operative utilization started (i.e. after the end of the warranty period).

Except for the persons accompanying the project and the client within the scope of maintenance and after-sales services (i.e. 2nd level support staff) the resources are released and become available for new projects. This also applies to material goods such as project infrastructure (project premises, hardware and software used for development and tests) in addition to the staff themselves (and finally also the project managers).

The project is (internally) finally settled to determine its end result.

The documentation and material created in the course of the project are centrally collected, archived and filed.

A project debriefing takes place comprising both the involved staff of the client and also from us (experiences, lessons learned, etc.)

All in all, this phase is kept rather short and also only requires minor cooperation from the client.

Configuration Management

Survey of the SAGA Configuration and Change Management

Configuration Management as how a project ensures that the contents of systems are known, that changes are authorized, and that the documentation accurately describes the system. Configuration management is both a management and a technical discipline that involves nearly everyone on a project either actively, as performers of configuration management activities, or passively, as recipients of configuration management information.

The essence of configuration management is control-identification of what needs to be controlled, controlling those items through change management and configuration control functions, reporting metrics to help understand the status of the items under control, and finally, auditing to ensure that the control processes are working effectively.

The SAGA Configuration Management comprises following elements:

- Version and build control-Version control provides unique identification of each version of a configuration item ad the ability to recreate any version, and should support sequential progressions (version B supersedes version A) and branching (versions C and D are concurrently valid in different scenarios). Build control ensures valid combinations of components and versions, from subassemblies to the entire system, as well as the ability to recreate any build.
- Baselines-Point-in-time collections, of specific versions of items, which become the reference points for subsequent change management.
- Change management-Controls to ensure that the system does not diverge from its baselines without authorization and that the system includes all authorized changes. The change review and approval process will not be artificially bureaucratic.

Change requests

Engineering method described below assumes that at the beginning of the project all requirements will be specified in document “Requirements Specification”. Any change in approved document should be processed as change request for which suitable change request procedure should be agreed between all concerned parts. Such changes can lead to changes in project plan. This must be also considered in change request procedure.

Meeting minutes

The Consultant shall keep minutes of each project meeting. Unless concerned part does not objects in writing form within two weeks after receiving the minutes content, the minutes will be deemed as accepted.

Configuration and Change Management Processes

The project needs to view CM as part of the overall planning process. The CM program will be revisited periodically and validated in response to major changes in the scope of the engagement. It will also be revisited in response to feedback that identifies CM-related problems to help eliminate the causes of the problems.

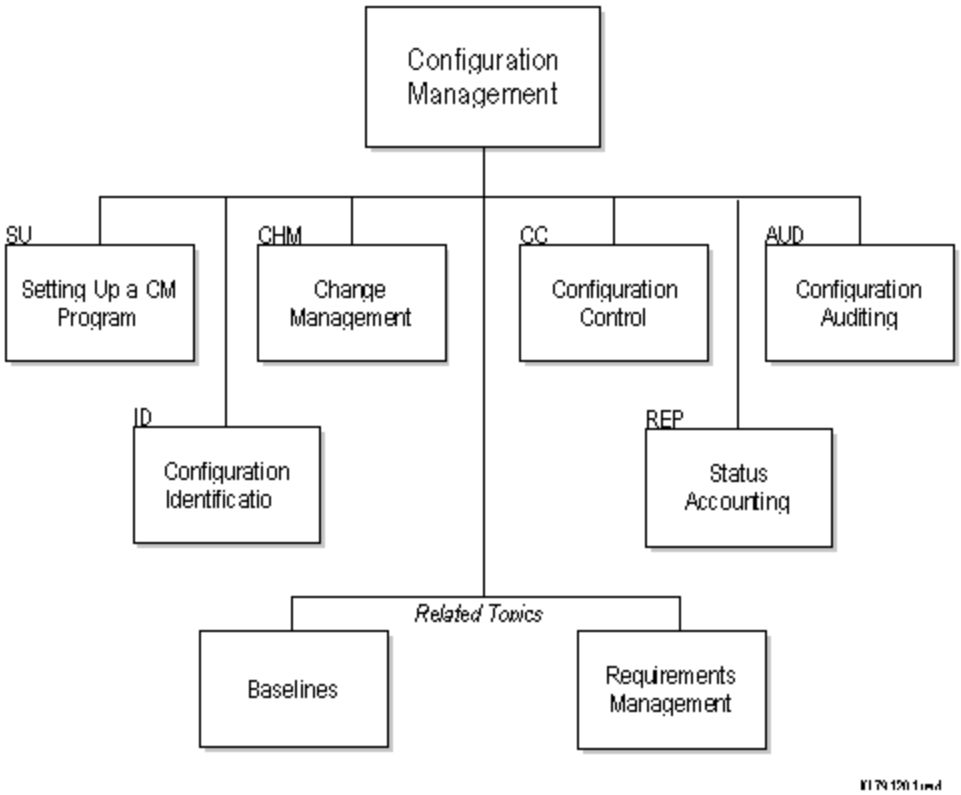


Figure 19 – Configuration Management Process

Setting Up a CM Program (SU)

Setting up a Configuration Management (CM) program comprises all the activities required to define the approach to configuration management on an engagement, including determining the activities to be performed; specifying the techniques, standards, procedures, and tools to be used; and assigning CM roles within the organization:

- Define the CM Process
- Define CM Data Collection and Reporting Process
- Specify Vendor and SubConsultant CM Requirements
- Select and Install CM Environment
- Set Up Repositories and Supporting Procedures

Configuration Identification (ID)

Configuration identification is the process of identifying the items to be controlled and documenting their characteristics. These items may vary considerably in complexity, size, and type. They may comprise all the hardware, software, and documentation for the business system. Characteristics may be documented in a variety of forms, including models, work products, plans, and reports that describe the business system being developed or maintained.

Configuration identification provides a hierarchical view of the system, showing a progressive decomposition into configuration items (CIs), components (anything between a CI and a configuration unit), and, at the lowest level, configuration units. Configuration units are the smallest elements that can be stored in a CM repository and have individual version control. For example, individual software programs and smallest replaceable hardware components are configuration units. In contrast, Cis provide a higher level view into system that is more suitable for providing meaningful management visibility and tracking.

The SAGA Configuration Manager will:

- Define Project Baselines
- Define CI Selection Criteria
- Select Configuration Items

Change Management (CHM)

Within the Change Management process the SAGA Project Management controls changes to the system to ensure that only authorized changes are applied. It involves receiving requests for change, analyzing them for impact and feasibility, approving them before implementation, and tracking them through completion. Key features of the SAGA change management are:

- A vehicle for proposing and tracking changes, commonly called a Change Request/Problem Report.
- An authorizing agent, representing major stakeholders, to approve or reject proposed changes. Examples of authorizing agents include program and project managers, and change control boards (CCBs).

Change is the most hazardous aspect of system development. Although the system design does not "wear out," its quality deteriorates over time as localized changes are superimposed on the original design, especially changes that are not implemented by the original architect and designer.

A successful change management process:

- Enables change decisions to be based on knowledge of the full impact of the change. This allows clients and management to understand both the cost and the benefit of a change and allows you to focus on changes that are necessary or that offer significant benefit.
- Ensures that all stakeholder interests are considered before changes are approved.
- Ensures that only understood and authorized changes are made, thus increasing the quality and maintainability of the system throughout its operational life.
-

Controlling the Configuration (CC)

The SAGA Configuration Management controls, manages, and maintains the historical trail for software (system or application, custom), documentation, hardware, work products, and any other documents (such as memos, correspondence, and distribution lists) that are important to project operation or as evidence of compliance.

At its most basic, configuration control consists of version control and build control. Version control entails identifying different versions of a configuration item, components, and configuration units, and maintaining their logical relationships (such as chronological sequence or domain of use). Build control entails integrating elements of the system and creating releases.

Key concepts of configuration control are version control, promotion, building configurations, and baselining configurations. Configuration control takes place within the context of the CM environment, which includes the repositories where current and historical information used to create builds and releases resides. For simplicity, we use the generic term “element” to signify a configuration item, component, or unit—the piece of the system that is under control.

The activities are

- Maintain Repositories
- Control Versions
- Create Builds and Releases
- Baseline the Configuration

Status Accounting (REP)

Status accounting consists of the set of activities associated with periodic reporting on the status of a configuration and the changes to that configuration. It uses data collected during other CM activities and during development and testing activities to present information about the current state of the system under development or in maintenance. Using information collected during all CM activities, CM

status accounting provides snapshots and reveals trends for change requests, configuration items, and CM activities such as builds.

CM status reporting includes metrics to support project tracking and estimating. The format and timing of reports varies according to the type of information being reported and audience needs.

Regular tasks are:

- Report Change Status
- Report Configuration Status
- Report CM Activity

Configuration Auditing (AUD)

By Configuration Auditing SAGA will confirm that what is agreed to deliver is what actually will be delivered to the Beneficiary, and that the CM practices have been followed. Configuration audits will be specified in close cooperation with the Beneficiary. Therefore scope and formality of configuration audits are directly related to the Beneficiary requirements and the size and complexity of the solution

The main tasks are:

- Conduct an Audit
- Respond to Audit Findings

Quality Management

Advantages of the SAGA Project Quality Management

An effective quality system reduces risk by increasing delivery consistency and efficiency. A consistent approach is easier to remember, measure, manage, and transfer to different situations; and if practices of proven worth are repeated, the risks associated with variation are minimized.

At the same time, a systematic, measurement-based approach to quality management identifies areas that work well and areas needing improvement. This increases confidence in identifying areas for reuse and in targeting improvements where the benefits will be greatest. An effective quality system also ensures that processes are consistently aligned with the goal of delivering products and services that meet the project's requirements.

Since SAGA is ISO 9001 certified, an effective quality system is already installed on which the quality management of the project can be based.

Elements of SAGA Project Quality Management

The SAGA quality management involves the following elements:

- SAGA Management accountability for quality;
- Clearly stated management goals for quality;
- Measurable objectives, based on the goals that can be used to track progress and achievement;
- A quality system, comprising the organization structure, roles, responsibilities, processes, procedures, standards, methods, techniques, tools, training, and resources needed to fulfill the goals and requirements;
- A quality system includes these capabilities specifically for quality management:
    - Identification and correction of defects in products and services. Corrective and preventive action to address problems in processes and other quality system elements.
    - Continual feedback and improvement within the quality system to increase its performance and productivity.
    - The capture, analysis, and use of measurement data to guide corrective, preventive, and improvement actions.
- The SAGA Management commitment to the quality system, including adequate resources for execution;
- Installation of the role of a Project Quality Manager.

Processes of the SAGA Project Quality Management

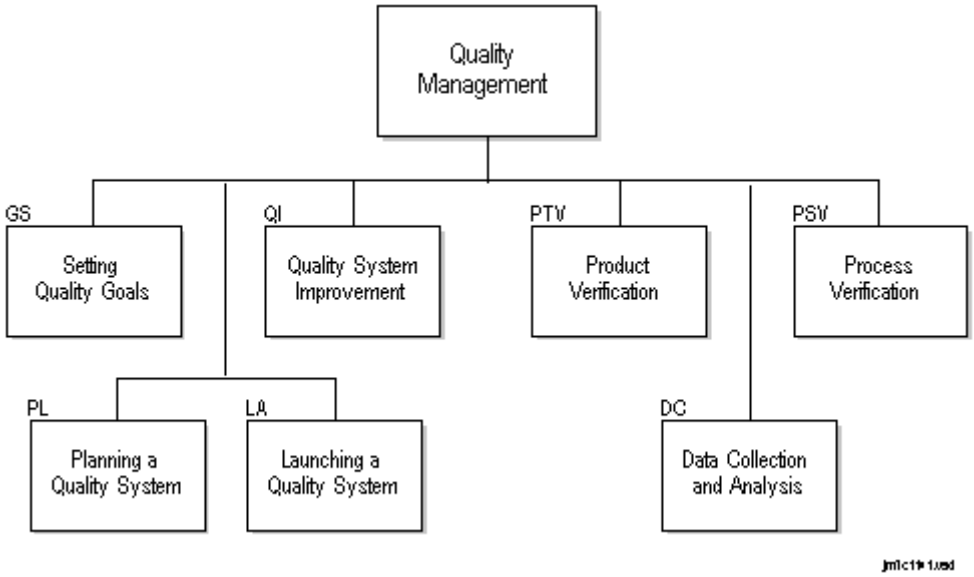


Figure 20 – SAGA Project Quality Management

The SAGA Quality management comprises the following processes:

Setting Quality Goals (GS)

Define goals for quality that are capable of being translated into measurable objectives. Goals are driven by the requirements of the organization, the nature of the product or service, business risks and expectations, and client needs.

Planning a Quality System (PL)

Define measurable objectives based on the goals, and a quality system that can meet the objectives and fulfill contractual requirements while capturing measurement data so that the level of quality can be monitored and corrections made as needed. The responsibilities, constituent elements, and operation of the quality system are described in a Quality Plan.

Launching a Quality System (LA)

Install quality system elements and bring the planned quality system into operation. During this process the commitment of Programm Management and the Project Implementation Commitee is made clear to all staff members and (if applicable) clients. Launching is done to establish a new quality system and can be repeated as elements are added or if major changes are applied.

Product Verification (PTV)

Apply techniques to discover defects or problems in work products, deliverables, and services produced by projects and lines of service; capture relevant measurement data for analysis; correct the discovered defects and/or apply corrective action to the quality system to prevent their re-currence; and communicate the results to management, staff, and (if applicable) clients. Some typical product verification techniques are reviews, inspections, and testing.

Process Verification (PSV)

Apply techniques to discover whether the planned quality system is being followed and whether it is effectively addressing current requirements; capture relevant measurement data for analysis; apply corrective and/or preventive action to the quality system to address the findings; and communicate the results to management, staff, and clients. Some typical process verification techniques are audits and assessments. (Corrective actions prevent the recurrence of previously encountered product defects and problems; preventive actions prevent the occurrence of defects and problems that have not yet been encountered, but that could occur.

Risk Management

Advantages of the SAGA Project Risk Management

The SAGA Risk Management helps the project to deal with uncertainty and some level of uncertainty is inherent in all projects. Even with careful planning, you will never be able to completely predict, prepare for, and control all project events. SAGA Risk Management will handle risks during the project to avoid adverse events and minimize their negative impact and suffer the consequences. The SAGA Risk management provides more options to direct the project away from possible negative outcomes and towards positive ones. The proactive Project Management of the Consultant anticipates and prepares for possible adverse future events, then selects an alternative action plan to enable project objectives to be achieved successfully. As a result, overall project success becomes much more likely.

Activities of SAGA Project Risk Management

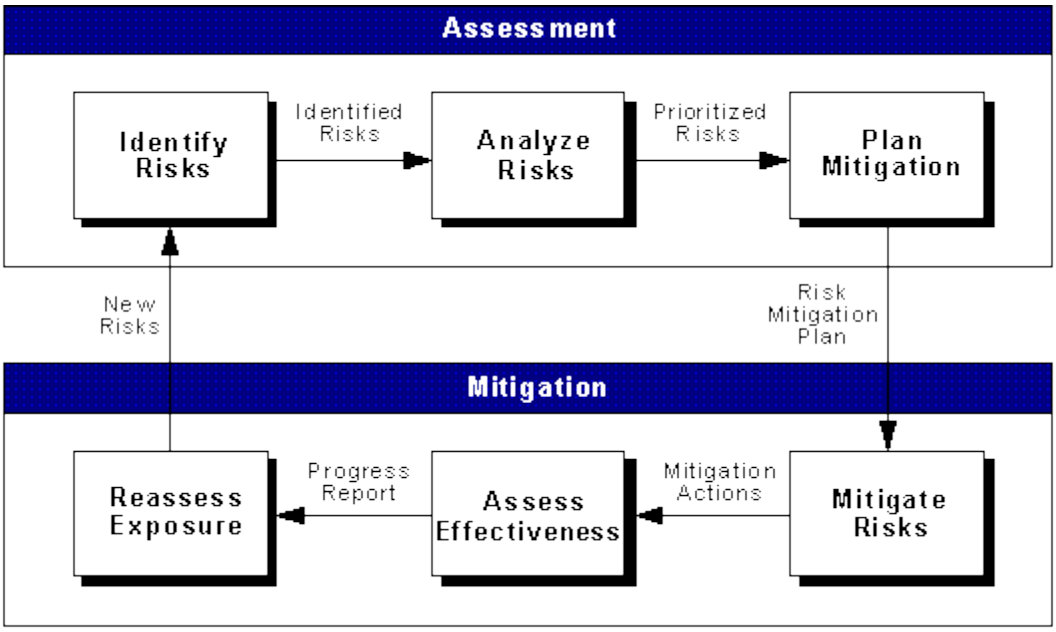


Figure 21 – Activities of SAGA Project Risk Management

Identify Risks

Since Risk identification is the foundation of the risk management process, the Project Management of both parties begin identifying risks during the proposal process and continue it throughout the project build phase. Risks will be found by reviewing and analyzing sources of information including the request for proposal, the proposal, the contract, and any notes, and will be documented in special Risk Management Worksheets.

Analyse Risks

The project management team will set priorities on risk management activities by analysing and quantifying the threat posed by each risk. Risk analysis helps determine the intensity with which the risk can affect the project and indicates what resources to apply to control or eliminate those risks. Risk analysis is an ongoing process, and any new or changed risks will be incorporated into the analysis from a project's start through to its completion.

Plan Mitigation

By planing risk mitigation to eliminate, reduce, or control project risks, the Project Management of both parties become a proactive player of the process.

Mitigate Risks

The Project Management team implements the Risk Mitigation Plan. It activates preemptive risk mitigation strategies. It implements, for example, a formal scope control process before the project experiences scope problems. It monitors the risk warning flags and reacts quickly to implement the planned direct risk mitigation strategies. It implements strategies in increasing levels of intensity. For example, it tutors a problem staff member before deciding whether to replace him or her. When necessary, it applys indirect risk mitigation strategies, such as extending work hours to reduce a schedule slippage.

Assess Effectiveness

The Project Management team will assess the effectiveness of its risk mitigation actions. At a low level of detail the status of each risk where a mitigation strategy is in place will be monitored. At a high level of detail the effectiveness of risk mitigation strategies will be controlled by tracking the cost, schedule, and quality dimensions of the project. Work product quality will be assessed through individual reviews and testing.

Reassess Exposure

Often late in the project a significant problem caused by an undetected risk is discovered. To avoid this situation the project's current risk status is periodically evaluated to address the dynamic realities of a project. This becomes more important as the project approaches integration and deployment. Frequently, new risks arise from integration issues. Therefore the Project Management team reassesses risks more often during the final three months of the project and during the first month of deployment.

Change requests

Overview

After contract award, modifications in terms of scope, functionality, time or quality of the project can only be requested and implemented by a formalised procedure which is called Change Request Procedure. The Change Request Procedure identifies, controls and documents modifications in the project.

The need for modifications may arise on one hand due to discrepancies in the fulfilment of the contract, these modifications are referred to as *problems*. On the other hand, the need for modifications may arise due to deviations from scope of work as agreed upon in the contract, these modifications are referred to as *changes*. Changes result in most cases in additional expenses which generally are born by the requesting party.

The Consultant shall keep minutes of each project meeting. Unless concerned part does not object in writing within two weeks after receiving the minutes content, the minutes will be deemed as accepted.

A Change Request (CR) refers to both kinds of modifications, i.e. a problem or a change. It can be applied for by both parties, i.e. the Beneficiary and the Consultant. Each CR shall be documented in a Change Request/Problem Report.

The Change Request Procedure is embedded in the Project Management Methodology **PMI**. The objective of Change Request Management is to reach a decision as quickly as possible on the requested modification, to permit Change Requests to infiltrate the project in a controlled manner, to process claims in a reliable way in terms of cost and time, to avoid contractual penalties and to document the deviations from the contract.

Scope

Each modification or deviation of defined scope, time (project schedule) or cost agreed upon in accepted proposal has to be made using this procedure. Request for change can arise either from the Consultant or Purchaser when the change is deemed necessary.

If the Consultant finds that a change request requires change of agreed cost and/or schedule, he will inform Purchaser and ask the approval of the request.

Definition

Change request	A formal project mechanism for tracking the progress and documenting the results of project management activity regarding a request for change.
Scope	The required set of verifiable products and services with specified characteristics that a project undertakes to provide to a client. Scope is established initially through the Project Plan.
Scope Change	A scope change requires an adjustment to the project work plan and nearly always impacts project cost, deliverables or schedule.
Scope Creep	The cumulative effect of allowing additional requirements after a project has started without considering the impact on project cost or schedule. Scope creep arises from the misapprehension that such small additions will not affect project scope.

Escalation	The transfer of management responsibility for approval of a change request to a higher level of decision-making for the project.

Change Management is an IT Service Management discipline. The objective of Change Management in this context is to ensure that standardized methods and procedures are used for efficient and prompt handling of all changes to controlled IT infrastructure, in order to minimize the number and impact of any related incidents upon service. Changes in the IT infrastructure may arise reactively in response to problems or externally imposed requirements, e.g. legislative changes, or proactively from seeking imposed efficiency and effectiveness or to enable or reflect business initiatives, or from programs, projects or service improvement initiatives. Change Management can ensure standardized methods, processes and procedures are used for all changes, facilitate efficient and prompt handling of all changes, and maintain the proper balance between the need for change and the potential detrimental impact of changes.

The Change Request Procedure

The Change Request Procedure generally starts after contract award. It is initiated when a modification is requested. In order to request a modification, the Change Request Applicant fills in the Change Request/Problem Report. The next step is the CR Screening which is done by the Change Control Board (CCB) in order to prioritise the CRs. The third step is the CR Analysis that results in an estimate of the effort of the CR in terms of technology, time, cost and a proposal of a solution. Thereafter, the CCB approves or rejects the CR. Having approved the CR, the CR is implemented, validated and controlled.

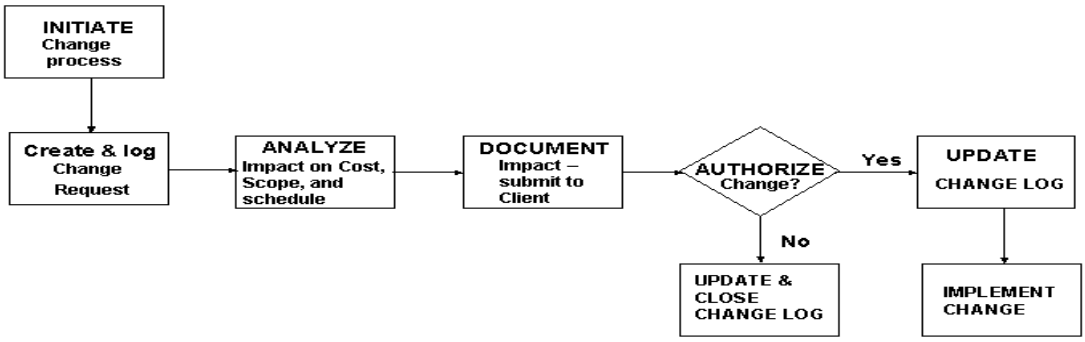


Figure 22 – Change Request Procedure

Change Request/Problem Report

To apply for a CR during the project period, the Change Request Applicant fills in a Change Request/Problem Report (see form hereunder). The request shall be described as clearly as possible and substantiated in the report. The focus of the description shall be on the problem rather than on the desired solution. The requesting party shall provide sufficient supporting material to reproduce a reported problem and to support the initiation, evaluation and design of CRs. The material may include reports and other documentation incidental to prioritizing, tracking, assigning, fixing, and closing out CRs. They shall be attached to a hard copy of the Change Request/Problem Report.

In case the Consultant requests a change, the Consultant shall include in the Change Request/Problem Report the effect of the change to the defined scope of work in terms of time and cost.

The report has to be signed by the project manager of each party. The CR is then sent to the CCB.

Change Request Screening

Having received the Change Request/Problem Report, the CCB approves or rejects the work on the CR. It is adviseable for the CCB to screen all submitted requests. The screening can identify emergency requests, determine whether the submitter has provided enough information for the analysis, recognise duplicates and interrelationships and classify requests. The classification of requests means the determination of whether the CR is a problem or a change. Finally, the CCB forwards the CR to the party that is responsible for implementing the CR.

Change Request Analysis

In case the CR is submitted by the Beneficiary, the Consultant shall provide a rough estimate of the effort and time required for analysis. This is called *impact analysis* and shall be documented in the Change Request/Problem Report.

In case the analysis requires an extensive examination as to whether and under what conditions the change can be carried out, an agreement on additional remuneration can be demanded. In case the analysis requires an interruption to the work in progress, the Consultant may demand remuneration for the period of interruption and/or an increase of the fixed price as long as the employees affected by the change cannot be assigned to any other work. As for the completion criteria, the completion period shall

be extended according to the working days, the contractual work has been interrupted, as well as a sufficient period to start again.

The Consultant shall then analyse the CR within a period prearranged by the Change Control Board to determine the impact of the change/problem in terms of time, cost and feasibility. In order to do so, the Consultant clarifies the technological consequences, consequences on the time schedule, the cost and on the alteration of contract. All results shall be documented in the Change Request/Problem Report. The analysis includes a determination of the root cause of the problem and a proposal of a possible solution.

The analysis contains the impact and risk in parallel with the estimation of effort, cost and time schedule. If more than one solution is proposed, the Consultant shall determine the technical, cost and schedule impact and risk of each.

In case the CR is submitted by the Consultant, the Beneficiary analyses the CR within a prearranged period of time and evaluates the effect of the CR as described by the Consultant in the Change Request/Problem Report.

After anaylsing the CR, all analysis results including the estimates and proposed solution are documented in the Change Request/Problem Report and forwarded to the CCB for review and disposition. The Change Request Applicant shall then send a notification on acceptance or rejection to the CCB a.s.a.p. but not later than 5 business days after the report has been sent to the CCB. The Consultant shall proceed with the work according to the existing contract as long as there is no agreement reached on the processing of a change.

Approval

Upon acceptance of the CR by the CR Applicant, the CCB evaluates the change/problem in order to decide on the outcome of it, i.e. the CCB accepts the implementation or rejects it. In order to do so, the CCB weighs the request in terms of cost, risk and business benefit. All CRs are documented with their complete derivation (history). The approval to implement the change is documented in the Change Request/Problem Report. All necessary alterations of contract are performed accordingly.

Change Request Realisation

The first step in this phase is to prepare an action list that includes all CRs. The CR then shall be scheduled for implementation and the project plan shall be updated accordingly. The specifications of the CR, the time schedule and other dependencies shall be documented in a meaningful CR report.

Change Request Validation and Control

The progress of the CR implementation is monitored and controlled until the change is closed. For the CR process, periodical reports shall be written. The validation and acceptance of the change shall be induced and after completion the Project Management of each party shall sign off the completion (hand-over and acceptance) in the Change Request/Problem Report. All results shall be documented in the Change Request/Problem Report log. Finally, all CRs shall be supplemented to the contract.

Status Monitoring and Reporting Procedure (CR.070)

This procedure defines how to monitor the Project status and how the Project progress reporting will be done. Project status monitoring is done by monitoring project progress and costs as well as by monitoring resolution of open issues, risks, problems and changes. Reporting on the project is done through the periodical report on project progress, as agreed within Project Managemet team.

Steering Committee meetings

Regular Steering Committee meeting should be held regularly (e.g. at least every four weeks) to review project progress and resolve outstanding issues. The Purchaser's Project Manager and the Supplier's Project Manager will report to a steering committee that will review project progress. Any issues that would affect the cost and the timeframes of the project would be addressed in these meetings. The Supplier's and Purchaser's Project Managers would jointly discuss and agree on the report for the Steering committee meeting.

Quality Assurance Procedure

The quality of products and work is maintained and upgraded within all the organization structures and locations of enterprise.

The most recent international standards recommendations for managing and enhancing the quality of conduct are in use. Therefore, upon the risk analysis, activity planning, periodical audits of work and results, quality is being managed and improved.

Because of the project-oriented entrepreneurship, Consultant has in place very strong and specific mechanisms of quality control and assurance. Quality of work and timely results are maintained by means of requirements of ISO 9001:2008. These mechanisms and work organization are recertified against the requirements of ISO, on a regular basis by the independent certification body.

The quality control is a comparison of the contracted measurable points with the contracted values. Based on the results of this comparison, the corrective actions are put in motion, as to attain the contracted level of quality. Consultant's quality department carries out periodical quality audits of the projects.

Quality assurance is a set of instruments by means of which we maintain the quality on the contracted upon level, maintained and upgraded through periodical audits. Some of the instruments of the quality assurance in use are: regular reporting about the project state, periodical Client - implementer meetings, periodical audits, managing risks and managing the corrective actions.

Accomplishment of this project goals, that is, the quality of the IS and service of implementation directly depend on the strict usage of the technical and professional methods. These methods assure quality through management of the audits and corrective actions. In the relation of Client's and project requirements, according to the ISO requirements, the implementer conciliates a quality plan for every segment which is being built and implemented.

Periodical quality audits of the projects in Consultant companies include risk management and quality of the delivery results. According to the ITIL recommendations, where appropriate, the checks are being conducted on:

- configuration management of all the elements that constitute the project or service: documentation, material and human resources, documentation and data, needed for successful conduct
- service desk, the assistance to the user during the implementation and product/service use
- incident management, by means of managing the user's requirements for resolving the delay of function of Consultant's products or service
- problem management, by means of managing the problems which originate from repeating incidents or the more complex incident which requires more steps to resolution
- change management, by auditing if any of the project or service components deviate from planned and contracted actions and resources in a project or a service
- delivery management is being audited as a set of actions and results of the project or service work. Documentation of the results and customer satisfaction is being checked
- service level management implies a comparison of delivered and contracted project or service results, and management of following corrective actions
- availability management implies the management of availability of physical and material resources needed, during the whole duration of a project or a service

These are just a few of the ITIL recommendations and short descriptions of how Consultant has implemented them in everyday practice. More detailed description is given in the Chapter **Quality Support Process**.

Configuration management

Based on analysis there is a need for establishing at least 4 instances:

- DEV: in the beginning this is a prototype instance which is used as application and data migration development environment. After go-live this instance is used for development of new or change of existing functionalities
- TEST: this instance is used for testing of configuration and extensions realized on DEV instance.
- PROD: in the beginning this is the key instance for functional configuration. After final data migration and go-live this is the instance used for everyday business processes
- TRAINING: instance used for training of end users, it's cloned from PROD to ensure that end users are trained in same environment that they will use afterwards
- PROD instance is never cloned from another instance, this instance is source for all other instances.

Changes from other instances are transferred to PROD instance only under following conditions:

- DEV: extension or modification is developed on DEV instance, after testing by end users on TEST instance and official confirmation, it is installed on PROD
- TEST: new functional setup is applied and tested on TEST instance with real data, after official confirmation setup is transferred to PROD

The Configuration Lab tool in the Oracle CR solution allows the definition of groups of data (both configuration changes and business data) that can be moved from one environment to another. This is a very effective tool for configuration and change management control.
The Configuration Lab provides the functionality to safely move data between various Oracle CR solution environments (e.g. between development, test and production environments). One valuable use of the lab is to experiment with changes to control tables. Users can add and update values in control tables, verify that the system behaves appropriately, and then move the new values into a test or production environment. An important benefit of this feature is the ability to synchronize an entire set of configuration-table values between two environments, thus facilitating configuration in a test environment and then moving the new values over to a production environment. The Configuration Lab makes use of a set of metadata control tables that define the relationships and rules for moving data. These same structures are also used in archiving.

Roles & Responsibilities

Change Control Board

The term Change Control Board (CCB) refers to a board with authority for reviewing and approving changes. It includes representatives of the project management from both parties, i.e. the Beneficiary and the Consultant. The CCB has the authority on the decision of a CR and to pronounce when a CR should begin.

The Change Control Board consists of the following roles:

- Programm Manager (Beneficiary);
- Team Members designated by the Programm Manager (Beneficiary);
- Project Director (Consultant);
- Team Members designated by the Project Director (Consultant).

Change Request Applicant

The Change Request Applicant requests for a modification in terms of scope, time or quality of the project by submitting the Change Request/Problem Report to the CCB.

Workproducts

Change Request/Problem Report

A Change Request/Problem Report contains a request for a modification in the Statement of Work. The Change Request/Problem Report is a form. Because this form is typically the vehicle by which a request is tracked, it includes fields for all information about request status from submission to closure. (please refer to the sample form hereunder)

Change Request/Problem Report Log

The Change Request/Problem Report Log stores information about a project's Change Requests Reports for tracking and reporting purposes. The Log is typically the project's database of all submitted Change Requests Reports.

Escalation procedure

In order to ensure a proper and efficient progress of the project work the Consultant proposes the following escalation procedure:

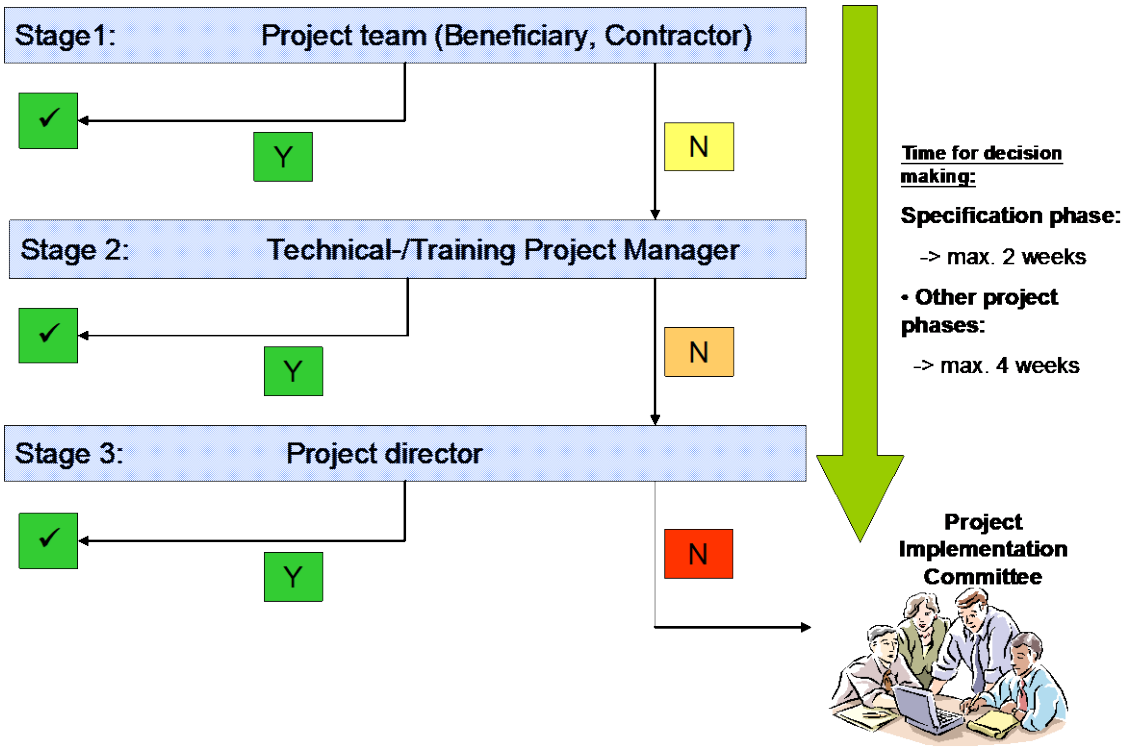


Figure 23 – Escalation Procedure

A check mark and a "Y" (for "Yes") means that a decision can be reached on the respective level, while a "N" (for "No") means that the decision has to be passed to the next escalation stage.

Project Organization

The organization of the project

Organizational Definition

The project management structure is based on a Customer/Supplier environment where:

- a Customer, Ministry of Agriculture, Forestry and Water Economy sector in the Republic of Serbia, specifies the desired Outcome, make use of the Outcome and pay for the project;
- A Supplier, provides the resources and skills to create that outcome.

Establishing an effective organisational structure for the project is crucial to its success and each project needs a flexible organisation structure flexible and is likely to require a broad base of skills for a comparatively short period of time.

In order to establish a professional project management structure, a Supplier must define roles and responsibilities that bring together the various interests and skills involved in, and required by, the project.

Good project management practice requires the fulfilment of a number of generic, well-defined roles. For the project to be successful it is important to define these roles at the outset.

A project management structure is a temporary structure specifically designed to manage the project to its successful conclusion to meet the requirements defined in the Project Brief.

The structure allows for channels of communication to decision-making forums and should be backed up by job definitions, which specify the responsibilities, goals, limits of authority, relationships, skills, knowledge and experience required for all roles in the project organisation.

Responsible Body

The institutional frame work of the project includes Ministry of Agriculture, Forestry and Water Economy sector and Contracting Authority, Republic of Serbia. Ministry of Agriculture, Forestry and Water Economy sector, as the Beneficiary will be responsible for the technical implementation of the program under the overall supervision of EC. The Republic of Serbia is the Beneficiary for the Project, and is responsible for the administrative and financial implementation of the project. The Contracting Authority has the ultimate responsibility for the correct use of funds and is responsible for the tendering, contracting and payments.

Facilities Provided by the Beneficiary and Other Parties

The Project team will be provided by the Beneficiary with a meeting room for joint activities and meetings. All relevant and existing information to support the project activities will be provided by the Beneficiary as well.

Facilities Provided by the Supplier

The consultant experts will be adequately supported and equipped. In particular, there will be sufficient administrative, secretarial and interpreting provision to enable experts to concentrate on their primary responsibilities. We will also transfer funds as necessary to support its activities under the contract and our employees are paid regularly and in a timely fashion.

The arrangements will allow for the max flexibility in the project implementation.

Equipment

No equipment will be purchased on behalf of the CA/Contracting Authority Country as part of this service contract or transferred to the CA/Contracting Authority Country at the end of this contract. Any equipment related to this contract which is to be acquired by the Beneficiary country will be purchased by means of a separate supply tender procedure.

Location

The location of the project is the IT Department and Ministry of Agriculture, Forestry and Water Economy sector of the Republic Serbia and any other locations where it may be appropriate for Supplier's staff during deployment periods.

Commencement date & Period of executrion

The period of execution of the contract will be as defined in Project time schedule..

Project management organization, Roles and Responsibilities

Saga proposes the following project structure, roles and responsibilities:

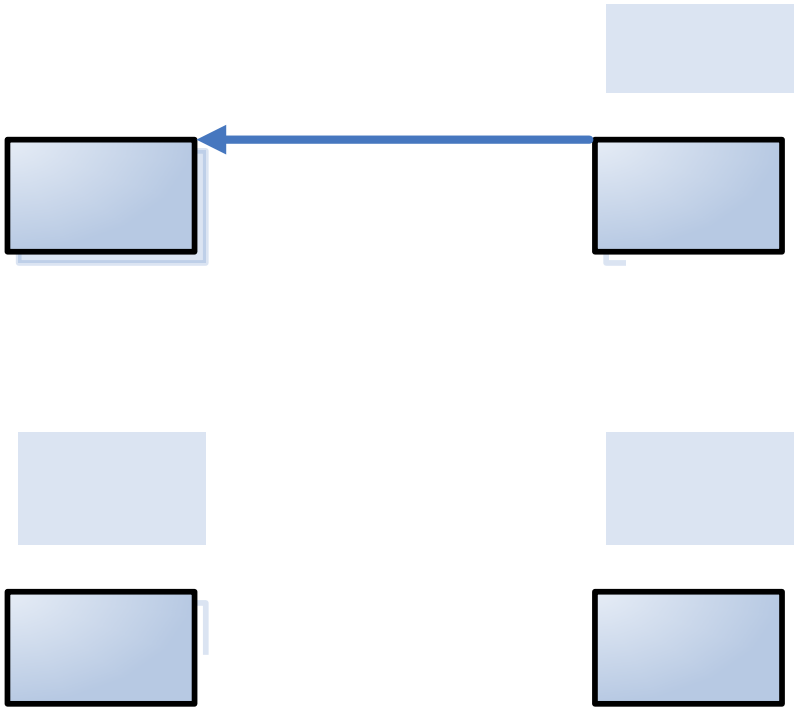


Figure 24 - Project Organisation

Arrows represents reporting paths.

The project management structure is the following:

- MMS: Ministry Management Services is composed of all the necessary roles and responsibilities in order to control the performances of a Supplier on the project. It assumes a number of roles who are represented with the table below.

Function	Name	Backup	Roles
Project Director	Director of Business Custom Directorate	To Be Identified	

Function	Name	Backup	Roles
Deputy Project Director	Head of IT Department	To Be Identified	
Head of Related Departments	Head of Related Departments	To Be Identified	
Team Leader	IT project Manager	To Be Identified	Executes of the IT activities described in the PQP Controls the IT activities Controls and manages the IT project requirements Verifies the progress and makes corrections if necessary Controls the execution of the IT activities described in the PQP Defines the roles and responsibilities of the IT experts and their relation with the business team.
Quality Assurance Manager	Quality Manager (To Be Identified)	To Be Identified	Verifies the PQP Elaborates the quality criteria for all major deliverables Participates in the acceptance/integration testing of the system Participation in the final verification and test phases

Function	Name	Backup	Roles
			Supports the project in using the applied methodology and best practices Recommends development standards and tools
Business Project Manager	Business Project Manager (To Be Identified)	To Be Identified	Manages and coordinates the business and functional tasks described in Project Plan Controls the business and functional processes Manages the business and functional project requirements Verifies the progress and makes corrections if necessary Controls the execution of the business and functional activities described in the PQP Defines roles and responsibilities of the business experts and their relation with the IT Team
Business Experts	Business Experts (To Be Identified)	To Be Identified	Elaborates the user requirements cooperating with the IT project manager and the end users

Function	Name	Backup	Roles
			Verifies the current processes and practice; investigates the necessary changes Elaborates the organizational changes wherever it is necessary Based on the testing strategy, prepares the acceptance test for the interim and the final version of the product Elaborates the functional training programme for the end users Participates in the training of the end users if necessary Participates in the development of the helpdesk and the error handling procedures of the system Provides initial base and verification data Helps the Consultant to prepare the user manuals Participates in the elaboration of the migration/rollout strategy
IT Experts	IT Experts (To Be Identified)	To Be Identified	Closely cooperates with the Consultant in the

Function	Name	Backup	Roles
			development of the system Participates in active and consultancy manner in the preparation of the deliverables defined in the PQP.

- CMS: Saga (Supplier) Management Services is the single point of contact for project management and follow-up. It assumes a number of roles who are represented with the table below.

Function	Name	Backup	Roles
Team Leader	**Please see at section 2.5**	Deputy project Director/Deputy Team Leader	Please see at section 2.5
Deputy project Director/Deputy Team Leader	**Please see at section 2.5**	Team Leader	Please see at section 2.5
Quality (Assurance) Manager	**Please see at section 2.5**	Business Manager (Business Team Leader)	Please see at section 2.5
Business Manager (Business Team Leader)	**Please see at section 2.5**	IT Manager (Technical Team Leader)	Please see at section 2.5
IT Manager (Technical Team Leader)	**Please see at section 2.5**	Business Manager (Business Team Leader)	Please see at section 2.5

- Commission for the co-ordination of the projects of the IT systems : the commission will be established in the CAS. Its role is internal provision and overall coordination of the projects of the IT systems. The commission will be responsible for the high level management of the contract and will be chaired by the head of the Project Implementation Unit or his representative.

- Project Board (Steering Committee) : the purpose of this committee is to regularly review progress on the implementation of the contract, consider major problems and delays, agree and decide how they will be resolved. The meetings will be held according to an agreement between MMS and CMS, not less than once per quarter and at the end of each important phase of work or whenever necessary. The project steering committee will meet every three months and at the end of each important phase of work or whenever necessary. The agenda for the meeting will be prepared and agreed upon by both parties at least one week (5 working days) before each meeting.Project Manager: together with his/her counterpart the Project Manager will be responsible for resolving all day-to-day management issues. Issues that cannot be resolved at the Project Manager's level will be referred to the Project Director. Any issues that may impact upon milestone dates will be reported at the Project Director Level.

    - Team of Experts: are all the experts identified and proposed by Saga in order to deliver the requested services:
        - Technical assistance:
            - Project Director
            - Deputy Project Director
            - Team Leader
            - Quality Assurance Manager
            - Business Manager
            - IT Manager
        - Trainings:
            - Training managers

Saga team staffs are THE key assets of our offer.

The project managers will be the main point of formal contact for day to day project issues (e.g. correspondence regarding deliverables). Senior representatives from us and Project Director of the CAS will meet to decide on major functional issues concerning the Contract which have not been resolved at the lower level. The quality managers will ensure compliance with the contents of the PQP and organise QA of all deliverables.

Security Management

Information is an asset that, like other important business assets, is essential to an organization's business and consequently needs to be suitably protected. This is especially important in the increasingly interconnected business environment. As a result of this increasing interconnectivity, Information is now exposed to a growing number and a wider variety of threats and vulnerabilities. Information can exist in many forms. It can be printed or written on paper, stored electronically, transmitted by post or by using electronic means, shown on films, or spoken in conversation. Whatever forms the information takes, or means by which it is shared or stored, it should always be appropriately protected. Information security is the protection of information from a wide range of threats in order to ensure business continuity, minimize business risk, and maximize return on investments and business opportunities. Information security is achieved by implementing a suitable set of controls, including policies, processes, procedures, organizational structures and software and hardware functions. These controls need to be established, implemented, monitored, reviewed and improved, where necessary, to ensure that the specific security and business objectives of the organization are met. This should be done in conjunction with other business management processes.

ISO/IEC 17799:2005 establishes guidelines and general principles for initiating, implementing, maintaining, and improving information security management in an organization. The objectives outlined provide general guidance on the commonly accepted goals of information security management. ISO/IEC 17799:2005 contains best practices of control objectives and controls in the following areas of information security management:

- security policy;
- organization of information security;
- asset management;
- human resources security;
- physical and environmental security;
- communications and operations management;
- access control;
- information systems acquisition, development and maintenance;
- information security incident management;
- business continuity management;
- Compliance.

The control objectives and controls in ISO/IEC 17799:2005 are intended to be implemented to meet the requirements identified by a risk assessment. ISO/IEC 17799:2005 is intended as a common basis and practical guideline for developing organizational security standards and effective security management practices, and to help build confidence in inter-organizational activities.

But how and where we start is that as ISO IEC suggests, we will begin by identifying CAS organization's information security needs and requirements. We will identify CAS security needs and requirements in the following way:

1. Perform a risk assessment. Identify major security threats and vulnerabilities. Then determine how likely it is that each threat and vulnerability will cause a security incident. Then evaluate the potential impact each incident could have on CAS organization given CAS overall business objectives and strategies. This will help us to pinpoint CAS organization's unique information security needs and requirements.
2. Study your legal requirements. Study all the legal, statutory, regulatory, and contractual requirements that your organization, its trading partners, Consultants, and service providers must meet. Look for all the information security requirements that must be met. This will help us to identify CAS organization's unique legal information security needs and requirements.
3. Examine your own requirements. Examine CAS organization's own information processing principles, objectives, and requirements. Study the information processing methods and practices that your organization has developed in order to support its operations. This will help us to identify and refine CAS organization's unique information security needs and requirements.

According to ISO IEC, CAS organization's information security program will be more successful if we accept the following suggestions:

- Make sure that the senior management visibly supports and is committed to your information security program.
- Make sure that the management has agreed to fund CAS organization's information security management activities.
- Make sure that CAS's approach to information security is consistent with CAS organization's corporate culture.
- Make sure that the information security policy, objectives, and activities reflect CAS organization's business objectives.
- Make sure that CAS organization understands its own unique information security needs and requirements.

- Make sure that CAS organization understands why risk management is central to your program and why a risk assessment should be performed.
- Make sure that CAS information security program is explained to all managers and employees and that they understand why it's important.
- Make sure that CAS distribute information that explains your information security policy and standards to all employees and other interested parties.
- Make sure that CAS provide appropriate security training, education, and awareness programs.
- Make sure that CAS organization establishes an effective information security incident management process.
- Make sure that CAS encourage people to provide feedback and to suggest ways of improving the performance of your information security program.

Make sure that CAS develop a balanced and comprehensive way of measuring the performance of your information security program.

Project Monitoring & Evaluation

There are various levels of control in the project. Most of the controls are:

- event-driven, including all the decision-making ones;
- Time-driven controls such as regular progress feedback.

At the project level there is overall control by the Project Board, which receives information from the Project Manager (and any assurance roles appointed) and has control over whether the project continues, stops or changes direction or scope.

The 'management by exception' approach will be applied meaning that having an approved Project Plan, the Project Board is kept informed through reports during all the project lifetime. The Project Board knows that the Project Manager will inform them immediately if any exception situation is forecast.

The major controls for the Project Board are:

- Project Initiation

- Inception Report
- End of Service Assessment
- Progress Reports
- Exception Reports
- Project Closure

The following Monitoring tools will be implemented during the project lifetime in order to insure the Project Board with the controls mentioned above. These tools will mainly be used by the Project Manager. All the information collected by these tools will be collected in order to be presented during the Monthly Progress Meetings up to a certain level defined in the Project Plan. The monitoring activities include progress monitoring, problem handling, issue management, quality control and acceptance procedures.

These tools are:

1. Service Terms of References

   Terms of reference is a mutual agreement under which the Consultant undertakes specific missions or tasks relative to the customer under specific conditions and constraints.

   Step 1: The CMS must write for each specific mission requested by the MMS, Terms of Reference (ToR) describing the mission of the consultant for a specific period

   Step2: The MMS reads and agrees on the ToR. This document is dated, signed by MMS and CMS and indicates the service start date and expected delivery date and a detailed specification of the service.

   Step 3: A copy of the signed ToR is kept by the CMS for archiving.

   The ToR will allow the project manager to ensure that the staff is delivering the services as specified by the MMS.

2. Meeting and Reporting

   - Kick-off meeting: the project will begin with a "kick-off meeting". The first meeting will define composition of the team, key points on methodology, initial set of questions
   - Project Steering Committee Meetings: this meetings will meet every three months and at the end of each important phase of work or whenever necessary.
   - Weekly team meetings and reports should be organized by the project management team. These meetings are the occasion to debate about low level observations and prepare the weekly tasks;
   - Monthly meetings will be realized between the MMS and the CMS to go through service delivery follow-up. Saga staff members could eventually be invited to sort out an escalated observation report;

- Inception Report includes a work plan for the experts for the duration of the project and a high level plan of the activities for the project duration. The report will provide overall evaluation of situation, include detailed work plan in terms of project and suggestions for including into the project any additional activities, necessary to archive expected results. The PQP will be part of the inception report.
- Final Report will be prepared by us. The draft version of final report will be submitted latest one month before the end of the period of execution of the contract. It shall reflect all project outputs covering activities carried out by experts, the progress achieved in relation to the project results, highlight any major problems which could arise during the implementation of the project, and make recommendations for follow-up.

3. Observation Report (OR) management

At any step, events might occur that could jeopardy the project success. In order to implement professionally a risk management process, the Project Director must have at its disposal, an information tool that will help to collect information about any disruptive project event.

Observation Reports can be written by any project party but are all centralised and managed by the Project Director.

- A party write an OR and transmits it to the Project Director;
- The Project Director assesses the risk related to the OR (Refer to the Risk Management Section). If the risk is HIGH, the management procedures remain the same but the problem is escalated to the Steering Committee and the solution is debated during the next the progress meeting or an ad-hoc steering committee meeting.
- The OR notification is logged;
- The Project Director proposes a solution to the OR reporter;
    - The solution is accepted,
    - If no solution can be found, the PD can escalate the problem to the Steering Committee;
- If the solution is accepted, an issue report is transmitted to the reporter and logged;
- The PD modifies the Project Plan accordingly to the implementation of the solution. If necessary, a contractual adjustment is proposed to the Steering Committee;

- The Committee confirms or not the contractual adjustment and/or the resources allocation for the solution implementation.
- When confirmation is received, the CMS informs respectively the PM and the consultant(s) of the decision to be implemented.
- After implementation of the solution, the OR is closed formally by agreement from the Steering Committee.

4. Contract and Service Delivery closure

Each services provided within the project must be formally closed. In order to perform a correct closure, two steps must be realised by the Project Manager:

- Step 1: the PD writes a Contract/Service Acceptance Note in order to notify all parties about the finalization. This notes must be formally signed by all members of the Steering Committee;
- Step 2: Service Delivery Debriefing Report written by the PD or Saga staff. This report will ensure that meaningful information is archived to achieve project knowledge persistence and transfer. This step is optional and will be specified in the Project Plan.

The individual roles and responsibilities are described in the ***Roles and responsibilities*** section above.

The activities that need to be performed at each phase of the project are described in ***Overview of phases and major activities*** section above.

Supplies & Provision of Materials & Responsibilities of the Beneficiary

This section specifies the supplies and responsibilities of the Beneficiary. It is a prerequisite for the success of the project that the Beneficiary provides the items mentioned below. The items should be provided in accordance with the timeline specified in the Project Plan.

Overall supplies and responsibilities

Beneficiary project staff

Project staff appointment

The Beneficiary will appoint project staff for the completion of the project, including but not limited to Programme Manager, Technical Project Manager, User Group Project Manager, Person in charge of acceptance, users / experts. This staff has to possess qualifications and skills required for the successful completion of the project. The Consultant has the right to reject a project staff member if he considers that the persons's qualification and skills do not match those required for the successful completion of the project. The Consultant has to make the rejection in writing and argument it. After the rejection is agreed with the Beneficiary Project Implementation Committee, the Project Implementation Committee appoints a new person.

Project staff appointment timeframe

The Beneficiary will announce to the Consultant within 7 days after the conclusion of the contract the names of employees, who will be responsible for the project on the part of the Beneficiary (Programme Manager, Technical Project Manager, User Group Project Manager, Person in charge of acceptance).

The Beneficiary will announce to the Consultant within 21 days after the conclusion of the contract the names of the rest of the employees, whose task will be to co-operate on the execution of the project (e.g. users / experts).

Work schedules

Beneficiary staff work schedules need to be specified. They have to clearly define each person's responsibilities to the project and the percentage of their work time dedicated to the project.

Staff absence

Beneficiary staff absence (holidays, vacations, etc.) need to be synchronized with the Consultant. When a Beneficiary staff member is off the job (including in case of illness, etc.) a competent person must be provided with no delay to perform his tasks and responsibilities.

Management

Programme Manager, Technical Project Manager, User Group Project Manager need to be appointed to work in cooperation with the Consultant team. Beneficiary project management staff has to be dedicated to work only on the specific project.

Support staff

Beneficiary staff who have responsibilities and/or activities related to the project will be made available to provide cooperation and/or perform tasks related to the project activities.

Over-time

Beneficiary project staff has to be available over-time if needed. The Consultant has to submit an argumented request to the Programme Director in order to be able to use Beneficiary project staff over-time. It is responsibility of the Programme Director to arrange the availability of the project staff.ž

Materials and Infrastructure

The Beneficiary has to provide the necessary hardware, standard software, and communication and network infrastructure (including Internet access, phone lines).

Infrastructure

The Beneficiary has to provide the necessary premises, access to office equipment for the purpose of the project (e.g. printer, copier, fax, telephone), electric power lines, etc.

Feedback

The Beneficiary has to provide timely feedback to questions from the Consultant arising during the project. Feedback should be provided:

- Immediately for questions, concerning day-to-day activities. If the responsible person is not available immediately, he/she should provide feedback as soon as he/she is available, but not later than the end of the same business day.
- Within 3 business days concerning questions of strategic importance for the project or questions requiring analysis on part of the Beneficiary). An extension of this term may be requested by the Beneficiary if there are reasonable grounds for that. The request and the argumentation have to be submitted to the Consultant's Project Director in writing at latest by the end of the next business day after the the question had been asked.

Cooperation with third parties

For the successful completion of the project the Beneficiary has to ensure the cooperation with all relevant third parties / external organizations.

The Beneficiary shall also ensure access to existing electronic databases and other systems that may be relevant to the project.

Ensuring the cooperation of third parties / external organizations and access to systems includes providing of consultancy, documentation, data, information, database tables, database structures, source code, operational assistance, and others that may facilitate the completion of the project.

Agreement on project methodology

The Consultant will provide the Beneficiary during the initial Analysis phase of the project with an explanation of his project methodology. It will be discussed with the Beneficiary and the Beneficiary will commit in writing to adhere to this methodology when providing any co-operation in this project.

Documentation and other information

Project documents templates

In the kick-off meeting of the project managers, the Beneficiary and the Consultant shall approve templates of project documents. The templates can be modified only with the approval of both parties.

Documentation

All documentation that is relevant to the implementation of the project shall be provided by the Beneficiary to the Consultant.

Information

Any relevant information about products / services to be supplied by third parties concerning application software or target environment of application software should be passed to the Consultant.

The parties shall on a continuous basis supply each other with vital information pertinent to the contract subject. If any of the contractual parties becomes aware of facts, that concern or might endanger the delivery of the Contract subject (e.g. expected changes in legislation, etc.), they are obliged to inform in writing without any delay the other party about these facts and the measures proposed.

Supplies and responsibilities needed for each project phase

For analysis

- Define a responsible person;
- Provide consultancy about all processes, functionality and configurations, which should be covered by the application software;
- Obligatory choice of proposed alternatives concerning proposed solution;
- Provide consultancy about interaction with external / third party systems, which should be covered by the application software;
- Define detailed description of interfaces to other systems.

For Design and Implementation

The following supplies and responsibilities are necessary in addition to the previously mentioned:

- Final information about network, hardware and standard software configuration;
- Special requirements concerning hardware and software (with respect to choice of solution alternative) used for development must be agreed between all concerned parties. Such special requirements (for example development on a target platform) can be fulfilled only with help of Beneficiary;
- Set of test data covering all use cases.

For Testing

- Defined network, hardware and software platform agreed with Beneficiary for acceptance tests;
- Defined network, hardware and software configuration agreed with Beneficiary acceptance test;
- The Consultant shall in the implementation phase prepare a test methodology for all components of delivery stating among other things that each acceptance test will be carried out at first in laboratory conditions and subsequently on a sample workplace agreed upon with Beneficiary. Testing scripts will be jointly defined;
- The Beneficiary project managers and users / experts prepare data necessary for execution of the acceptance tests and execute with support by consultants of the Consultant the acceptance test of the respective subsystem and its interfaces in accordance with the implementation plan.

For Deployment

- The members of Consultant's team are allowed to enter all his relevant premises as well as other places of delivery, if that is required by the task of the person;
- Defined access to Beneficiary's systems.
- Real-life data directly taken from database.

This cooperation must be kept in accordance with approved project plan so the intended milestones can be reached. Requirements mentioned above are related to the proposed milestones and project plan and are crucial for their compliance.

Acceptance Process

Delivery Acceptance

The Purchaser will receive notification about deliverable (document or Deliverable Receipt Form). The Purchaser is responsible for testing and reviewing the deliverable according to the acceptance criteria (acceptance criteria will be specified in detail project plan). If a deliverable is disapproved, a detailed description of why it was rejected should be given to the Consultant in writing within seven working days of deliverable receipt with documented errors and omissions of the deliverable in detail. All errors and omissions must be detailed in the first rejection.

Errors are categorized in the following way:

1 – Serious error (the system is out of function)

2 – Functional error (part of the system in not working correctly)

3 – Error for which workaround solution exists (has no influence on system functionality)

4 - User adjustments (has no influence on system functionality)

The Consultant will make a reasonable effort to modify deliverable and correct errors of the type 1 and 2. After the correction, the deliverable will be returned to the Purchaser for testing and reviewing as detailed above. At this point, deliverable should be approved/disapproved in writing within three working days of deliverable receipt (if both parties haven't agreed differently). Affirmatively e-mail from the Purchaser will have the same meaning as the personal sign. If no feedback is received after seven working days, the deliverable will be deemed accepted.

Application system will be deemed accepted if there were no errors of type 1, and less than 10 errors of type 2. For correction of remaining errors of type 2 the target date will be agreed. If the Purchaser does

not provide a list of errors in the above described way, the deliverable will be deemed accepted upon the expiry of the period for delivery acceptance.

After the first day of production use, Consultant will deem that the Application system is accepted by the Purchaser.

Phase Acceptance

During completion, phase deliverables are reviewed, approved and accepted by the Purchaser. Acceptance certificate for the phase completion is issued and signed by Purchaser. Completion of the phase is prerequisite for starting the following phase. Considering milestones, evaluation of the individual phase should be done within five working days of the phase end.

After delivery receipt, if no feedback from the Purchaser is received after seven working days, the deliverable will be deemed accepted. Phase is considered accepted if all deliverables planned for that phase are accepted. If both parties are agreed, some deliverables could be transferred to the next phase.

Project Acceptance

Project acceptance will be done as defined in the bidding document.

Technical Solution Proposal

Based on the methodological principles described in the Methodology section, in order to overcome the major issues of information system development, software development process is considered as a set of consecutive model transformations. Transformation transfers from the business (domain) up to the executive model on the target technology platform.

There are three basic issues inherent to information system design and development:

- Overcoming business system complexity
- Coping with permanent change inside system and system environment
- Enabling interoperability between heterogeneous models for system specification

Overcoming business system complexity

Business system complexity can be managed applying two fundamental methodological principles:

- breaking down software development process into phases, thus defining IS development lifecycle, and
- decomposing whole system up to the level of manageable components, thus defining IS architecture.

IS development lifecycle applied in Saga is agile and iterative. Function oriented analisys and object oriented design are used as methods for business system decomposition.

Coping with permanent change

Since IS is an executable model of the business system, permanent changes in business system itself and system environment reflect on IS as well. To cope with this issue, IS should be realized flexible enough, so it can adapt to functional, technological and organizational changes.

Most suitable way to achieve this goal is to develop IS based on IS Repository. IS Repository is information system upon information system, that is, IS Repository specifies business system in a whole. Custom developed IS Repository is based on IRDS (Information Resource Dictionary Schema) standard.

Enabling interoperability

Saga's methodology considers that software development process is a set of consecutive model transformations. Information systems are designed and developed using several different models. Each of these models potentially conforms to different meta-model. In order to provide Interoperability between heterogeneous models, Saga's model organization is based on MDA (Model Driven Architecture). MDA architecture is enabled through custom developed meta meta-model named Domain Description Model.

In accordance with the aforementioned, the appropriate architectural categories are defined, whose realization is the result of certain phases of the software development process. Each of these categories specifies appropriate system components and their interaction, such as the business model, the functions of the system, the way the components interact, their physical tiers and the like. Thus defined, categories of system architecture perceive and describe the entire system from different aspects, which are later reflected in facilitating the design, development and maintenance of IS.

Domain Architecture

Defining the domain architecture requires identification of the relevant parts of a complex system, i.e. target organization. The organization is a complex social-controlled entity, with defined boundaries, which works continuously to achieve certain goals. As depicted in Figure 1, the goals of the organization are fulfilled by executing business processes which require adequate resources. On the other hand, resources must be provided promptly, in organized and managed way in the corresponding elements of the organizational structure.

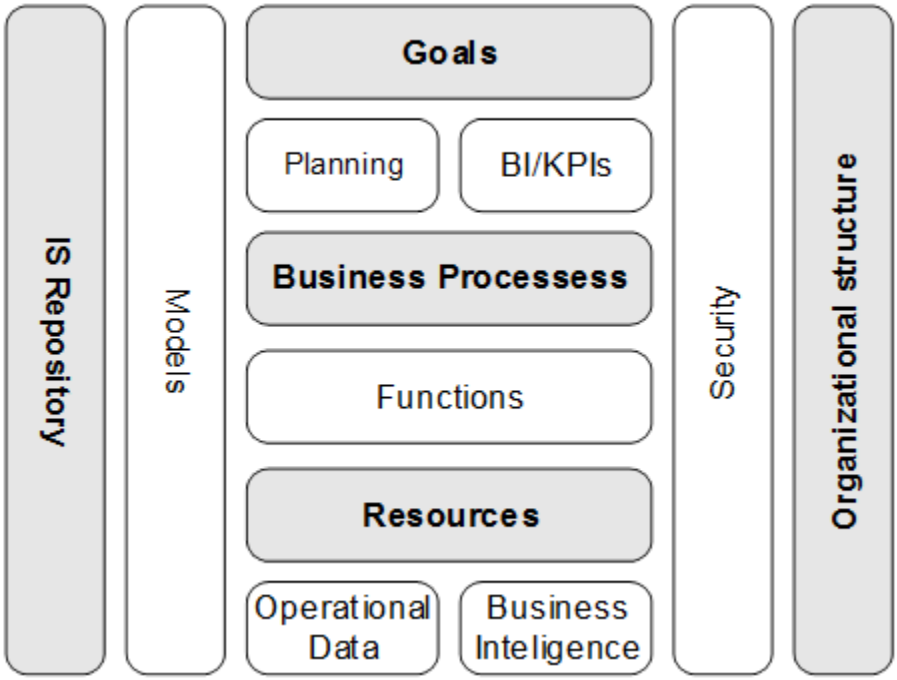


Figure 25 Domain architecture

The architecture is based on the Enterprise Architecture Framework, which is one of the most modern approaches to the development of complex information systems. In this approach organization's models along with its business and technical processes are coupled with models of the platform independent architecture.

Depicted architectural components (domains) are the essential building blocks which compounds the system. Each component may be analysed independently because it has no implicit semantics regarding concepts of other domains, increasing the reusability of the domain in the construction of different business systems. Therefore, depending on the complexity of the organization, but also on the foreseen target IS, domain architecture is the initial framework by which single components can be excluded with the possibility of their introduction in the later phases of the system extension.

Domain architecture components are determined at the Inception phase while defining the Scope of the project. Domain architecture will be realized through both, the Conceptual and Logical architecture, which can practically, on the basis of functional and nonfunctional requirements, be developed in parallel. In general, the Conceptual architecture is a version of Model Driven Architecture, while the Logical architecture is the realization of an appropriate domain(s) of the business system.

Therefore, Domain architecture is the basis of defining other architectural categories and a foundation for building of IS in general, since it is unburden of all organizational, technological and other constraints in which the system operates. Additionally, depicted Domain architecture may be used as pattern for any complex organization.

Conceptual Architecture

One of the main issues that arises in the IS design and development, and later in its maintenance, is mastering the heterogeneity of IS components caused by various meta-models by which components are specified and developed. Complex information systems are commonly realized using multiple meta-models (the relational model, object model, XML Schema ...). Furthermore, the models created in other phases of software development, particularly during analysis and design phase, are also specified using different meta-models (BPM, SSA, UML, E/R etc.).

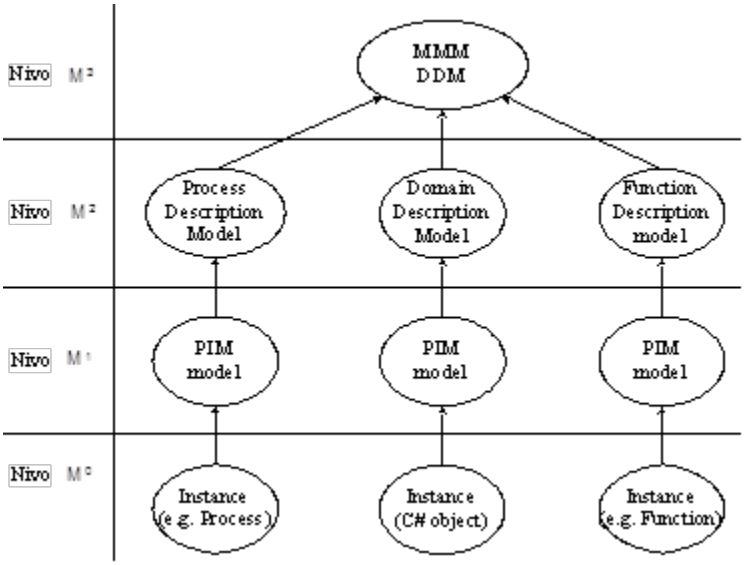


Figure 26 Conceptual architecture

In order to enable interoperability among heterogeneous models, as depicted in Figure 2, architecture realization is based on MDA (Model Driven Architecture) standard.

Considering aforementioned issues, Saga has developed Domain Description Framework - DDF, general software tool in support of information system specification and development. This software framework is composed of software components which aim to provide infrastructure layer in order to design and develop business system.

Presented framework supports Model Driven Software Development – the methodology that separates the specification of functionality from the specification of implementation of that functionality on specific technological platform. It also enables to generate executable code based on the system specification.

DDF is an core and integral part of majority of software products developed in Saga.

All DDF components are reusable and have belonging underlying model which specifies domain of interest.

As mentioned earlier, existing models are implemented in custom built application framework. Application framework is a set of general purpose software realized in S#/SQL Server implementation environment with the aim of providing a general infrastructure layer for the implementation of the concepts of a business domain. Designed and implemented in accordance with the MDA architecture, application framework basically involves proper metamodels, i.e. languages to describe the business processes, functions, system structure and other domains of interest.

This architectural category is ready to use, however business models that will participate in the construction of the FBOS system will be specified according to Tender documentation as described in Logical Architecture Section.

Software components that will be used in building of FBOS system will be depicted in more detail in Software Architecture Section.

Logical Architecture

The logical architecture is an integrated set of concepts that identify the behavior and structure of the system, as well as known constraints. The goal of this architecture is to identify and specify what system (IS) which is under development, should do. Similarly as domain architecture, logical architecture is considered without going into the details of the technology platform on which the system will be implemented.

Logical architecture contains three main parts:

- Functional model – composed of a system functions, as well as business processes that may include several functions in proper order of execution. This model defines the behavior of the system, possibly expressed as Activity Diagram and / or Structured System Analysis Model.
- Conceptual model - a set of system objects and their relationships, which represents the structure of the system. This model is specified as the UML Class Diagram or Entity/Relationship model.
- Constraints - a set of constraints that can't be expressed by model

By defining the functional and conceptual model, behavior and structures of IS under development are identified. It is necessary to define the relationship between concepts of different models (e.g. which functions are involved as activities in business processes or over which objects functions are performed).

This architectural category is the result of the Elaboration phase.

Logical architectural components are described in more detail in next section.

It is envisaged that the behavior and structure will be grouped into Logical Business Units and Common Business Objects. Logical units of work are functional units, i.e. the basic building blocks of Logical Architecture, while Common Business Objects are objects that are shared resources between multiple business processes or subsystems.

Logical architecture components

As briefly described in the previous section, the logical architecture identifies and specifies the behavior and structure of the system. In the case of complex organizations, behavior and system objects can be grouped into Logical Business Units and Common Business Objects. Logical Business Units are subsystems i.e. logical architecture building blocks, while Common Business Objects are objects that are used in multiple business processes or subsystems.

Common Business Objects

Common business objects are essential system objects that are most commonly used in multiple system business processes. The set of connected common objects forms a conceptual model of the system. Business objects are identified in the phase of detailed analysis and at this moment, the noticeable objects are:

- User - an object that has the permission to use application.
- Food Business Operator
- Food Facility
- Checklist
- Organization Unit
- Business Activity
- Food Type
- Retail Shop Type
- Activity
- Risk Criterium

All common business facilities will be identified in the Analysis & Design phase of the project. In addition to their identification, modeling of their structure and mutual connections will be performed, which will define the conceptual model.

Functional Units

It is envisaged that the system will consist of 8 modules. Functional units (with associated identified functionalities) are:

- Authentication, Authorization and Audit Module
- Food Business Operator Managing Module
- Food Facilities Managing Module
- Inspection Managing Module
- Public Portal Module
- Learning Module
- Reporting Module
- Reference Data Module
- Integration Module

Authentication, Authorization and Audit Module

This module implements precisely specified authentication and authorization system. The system will be implemented as a role-based web platform with different set of permissions based on the roles within the system.

Authorization management will be implemented within the Information system on the level of particular records in the database based on principles of ownership of a record. Only an owner of a particular record has a possibility to make changes on a record in certain situations, and a possibility for read only access has a user with a workplace that is on the same or higher organizational level, assuming that an user role allows access to related function or view of the Information system.

The Module will include following functionalities according to the RFP:

- User creation – System enables adding new users and setting their general information.
- User management – It is possible to view and edit user information, activate and deactivate user, set or reset user password.
- Searching users – One can query, filter, sort and list users.
- Role management – It is possible to grant and revoke roles to user.
- Audit - Changing and logical deletion of records in the database of the Information system is carried out by a history concept (a record in the database is qualified with an owner attribute and a time period of the record validity).

All details will be specified during the phase Analysis & Design of the project.

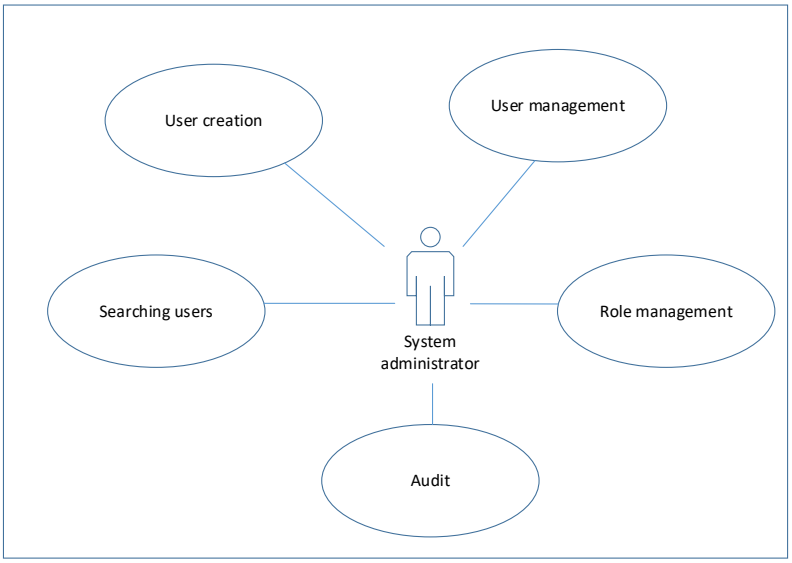


Figure 27: Authentication, Authorization and Audit Module functionalities

Food Business Operator Managing Module

This module allows food business operators (FBOs) to create a profile, select the industry and enter or update basic data, all of which is followed by verification process. After that, FBOs are allowed to notify/register business activities and important changes in activities (cease of activities, change of activities).

The Module will include all features needed for proper functioning of the system:

- Food Business Operator Creation – FBOs are provided with the possibility to create a new profile. Access is allowed only with a password which only registered businesses in the Republic of Serbia can get upon registration at the inspection in charge.
- Food Business Operator Modification – Once registered and with profile created, user can modify entered and add missing data.
- Food Business Operator Management – Once the FBO creates a new profile, the inspection in charge gets a notification that a new profile request has been submitted and waiting upon verification. The administrator verifies that the information has been properly entered and either accept the new profile or reject it in case of incomplete data request. The administrator is also able to deactivate FBO when needed.
- Searching and previewing FBO data – It is possible to query, filter, sort and list FBOs.
- Food Business Operator Classification - Once the FBO enters data, based on the classification the system will mark them as agriculture, veterinary or both.

All details will be specified during the phase Analysis & Design of the project.

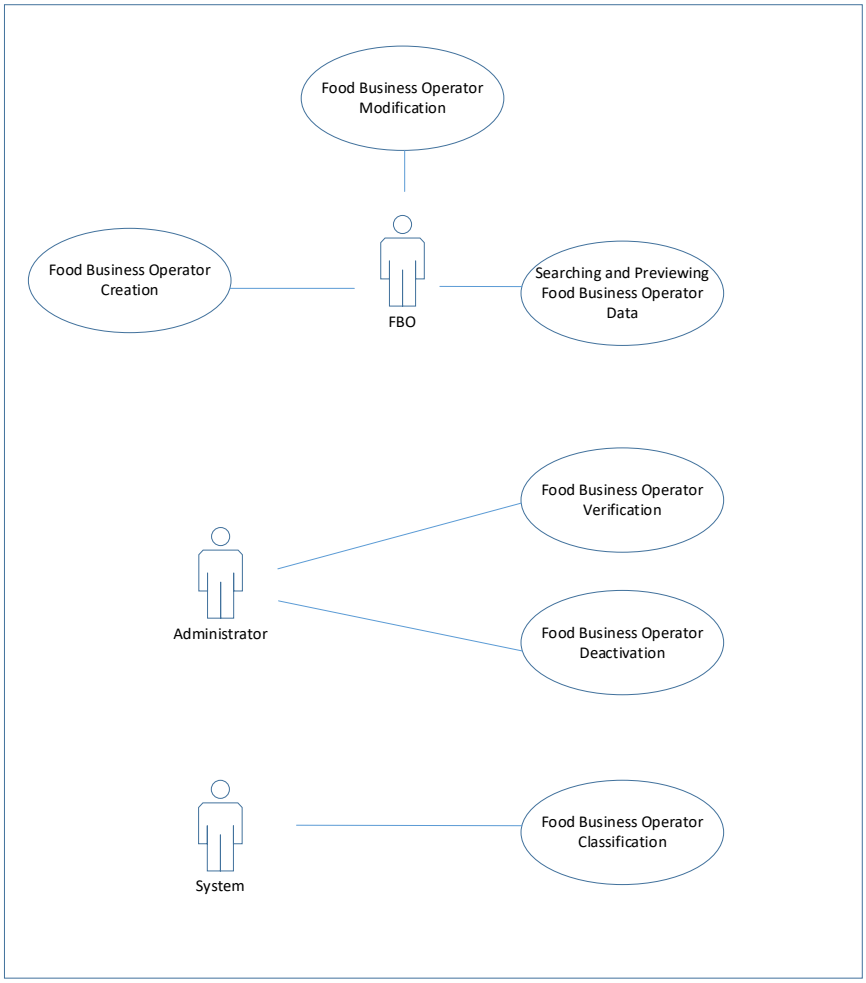


Figure 28: FBO Managing Module functionalities

Food Facilities Managing Module

This module allows FBOs to create one or more facilities, used for production, distribution and/or storage of food and feed.

The Module will include all features needed for proper functioning of the system:

- Food Facility creation – FBOs are provided with the possibility to create a new facility, filling all the required data.
- Food Facility modification – User can modify entered and add missing data of the created facility.
- Food Facility management – Once entered in the system, Food Facility can be activated or deactivated.
- Searching and previewing Food Facility data – It is possible to query, filter, sort and list Food Facilities.
- Food Facility withdrawal – System displays sign *Self-checked facility* for all those who performs regular self-checking, but also has a possibility for the sign to be withdrawn upon expiration of validity period.

All details will be specified during the phase Analysis & Design of the project.

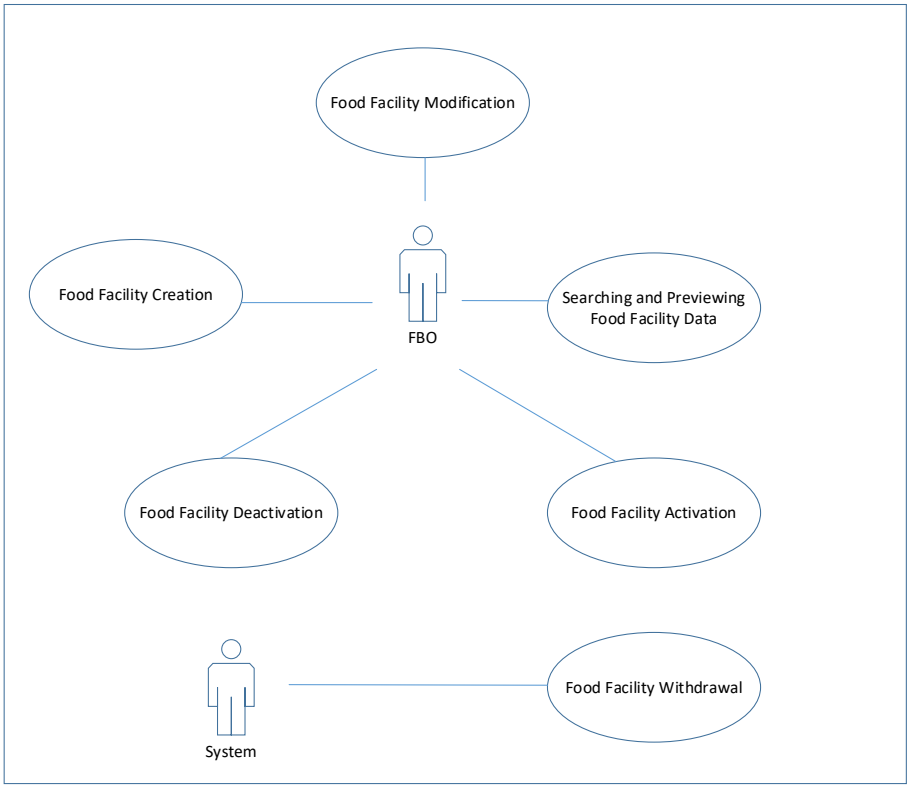


Figure 29: Food Facilities Managing Module functionalities

Inspection Managing Module

The system allows users (FBOs) to fill every 12 months electronic self-control checklists in line with inspection plans and oversight policies defined in the relevant inspection. The aim of the checklists is to help businesses assess their own system and identify gaps, as well as to provide the inspection with updated data regarding FBOs, their food safety management systems and their compliance with legal requirements.

Checklists consist of predefined questions with the simple yes or no option. Users are able to enter additional comment for each question. The system administrator is responsible for the content of checklists.

FBO receives a notification prior to the expiry of 12 months in order to be notified that it should fill in self-control checklist. Nevertheless, it is possible for the FBO to perform the self-control more frequently.

The Module will include all functionalities needed to fulfill the requirements of the RFP. All these functionalities can be divided into two groups:

- Checklist Definition
    - Checklist Creation – The system administrator creates a checklist, with the set of questions needed to be answered during the process of self-control.
    - Checklist Modification – Before activating the checklist, it can be modified.
    - Checklist Activation – Once the system administrator is satisfied with the content of the checklist, he can activate it, so that FBOs can use it for self-control.
    - Checklist Deactivation – When there is a need to add or remove one or more questions from a checklist, the previous version of checklist should be deactivated. Deactivated checklists can no longer be used.

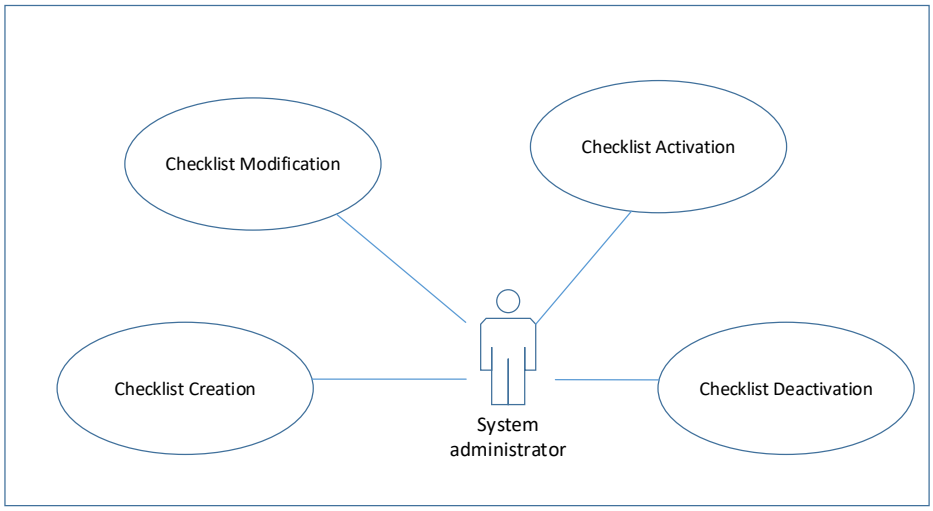


Figure 30: Checklist Definition functionalities

- Checklist Usage
  - Checklist Filling – FBO fills in checklist during the process of self-control, which needs to be performed every 12 months or in shorter period.
  - Checklist Submition – After answering all the questions contained in the checklist and, optionally, entering comments to them, user submits a result.
  - Checklist Preview – User can take a look at previously filled and submitted checklists.
  - Checklist Notification – System notifies FBO prior to the expiry of 12 months, as a reminder to fill in its self-control checklist.

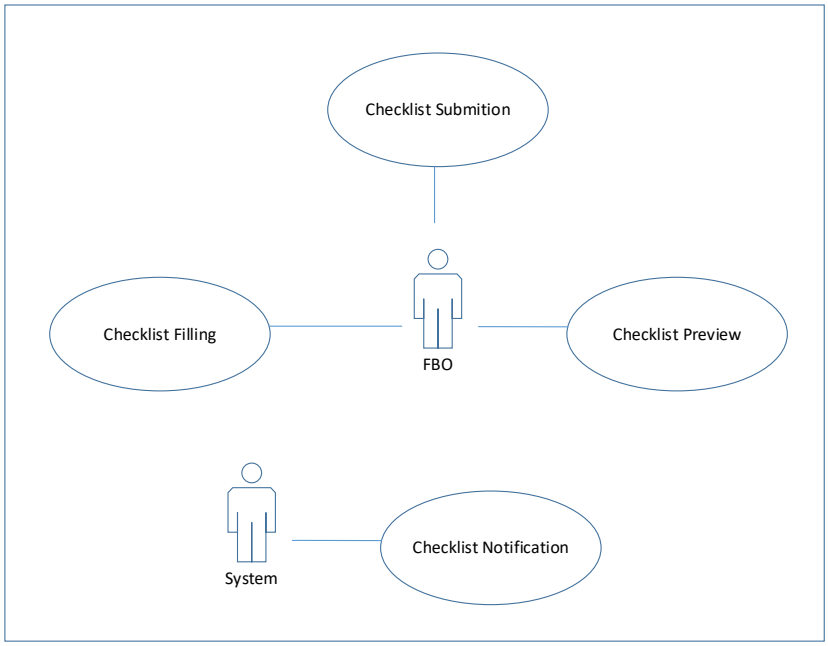


Figure 31: Checklist Usage functionalities

Public Portal Module

It is of fundamental importance for the general public to have access to the system. It is not possible for them to alter any of the entered data, but can primarily use the system as a tool for knowledge sharing and enhanced transparency.

The part of the platform open to the public contains variety of possibilities, such as:

- Display risk level of facilities based on the official risk categorization to the public.
- Display a HACCP model for retail facilities and instructions for use that FBOs may use to develop their own HACCP systems.
- Display guidelines for FBOs how to implement GMP and GHP.
- Display guidelines how to implement traceability in different types of food businesses.
- Display the list of all registered facilities from the database.
- Display sign (icon) *Self-checked facility* for all those who perform regular selfchecks with possibility for the sign to be withdrawn upon expiration of validity period. The display sign is automatically posted for the facilities that perform regular self-checks. If the facility doesn't perform the regular self-check, the system automatically withdraws the sign.

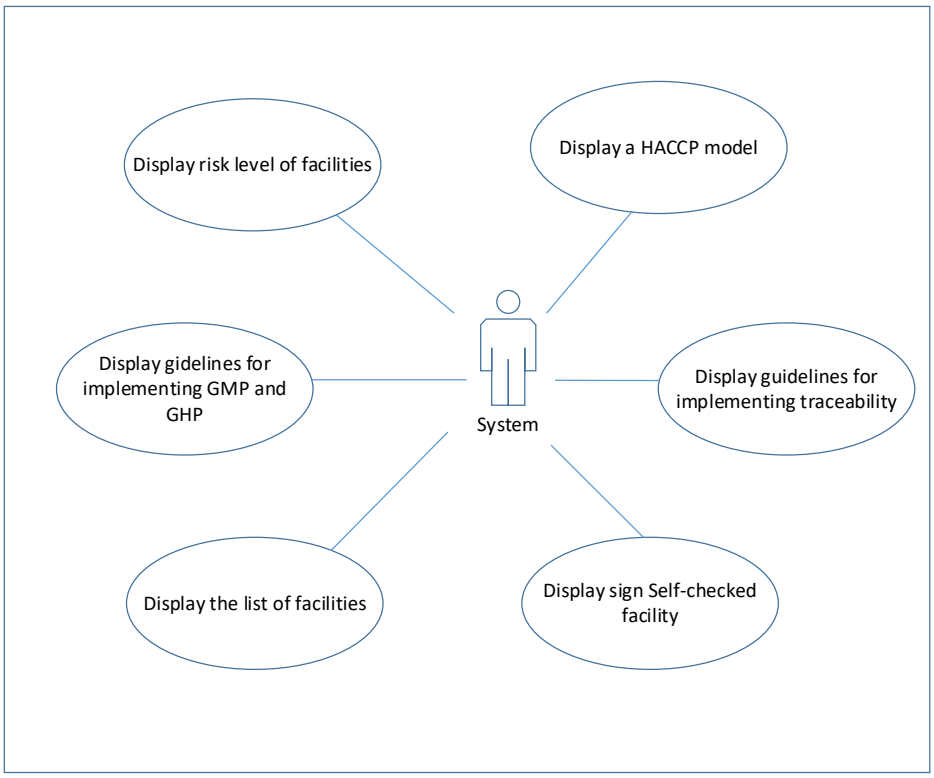


Figure 32: Public Portal Module functionalities

Learning Module

Learning module provides FBO and general public with how-to guides, rulebooks, required laws, articles etc., on various aspects of key topics.

Learning content can take several formats, which can be controlled by the system administrator. Content can take various formats:

- A post / text added by the Administrator, that can involve formatted text as well as embedded media, separated into different sections. This is achieved by posting pdf documents or using a rich text editor.
- An external link, that is, a URL to an external site.
- A media file, such as a video or an image. Videos should be hosted on Youtube and be uploaded there before they are added to the learning module.

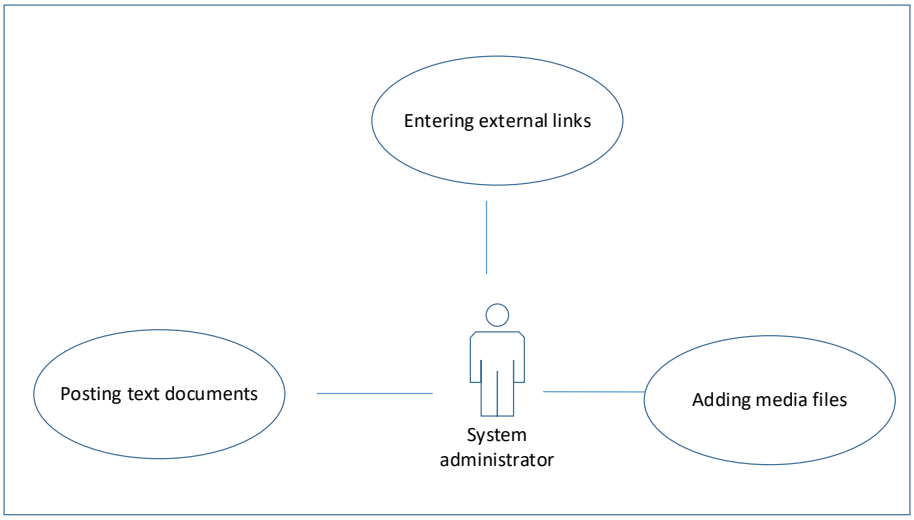


Figure 33: Learning Module functionalities

Reporting Module

Information system enables creating statistic reports. Core of reporting module is dynamic management and grouping of realized general and specific attribute values of different document types and grouping of results according to entered data as well as filled self-control checklists.

Reports can be generated in form of tables and diagrams, as well as exported to xlsx. In reports creation, there is an option of giving the different criteria such as: time frame, inspection, type of object, object size, risk criteria, product type, classification, location, general or specific attribute.

Examples of reports that might be able to generate is list all facilities: in specific part of Serbia, in retail, with more than 50 employees, with high risk.

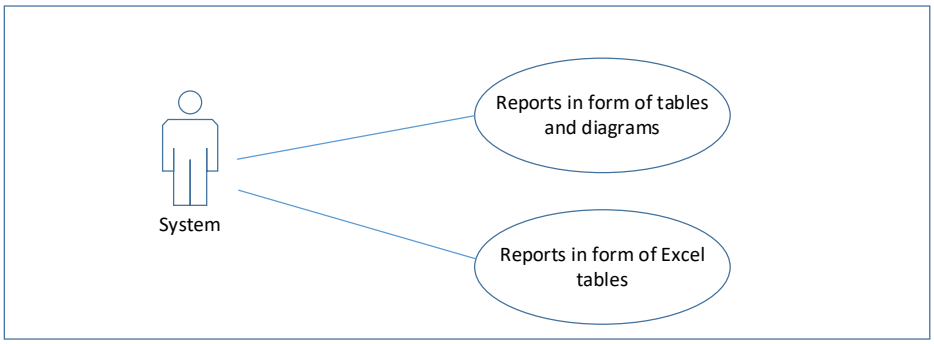


Figure 34: Reporting Module functionalities

Integration Module

Integration module will be used to exchange data between the Serbian Business Registry Agency information system and the FBOS through the standard web services implementing SOAP protocols. During the first takeover data will be taken via an Access file and the subcontractor has to enable importing the data into the system periodically via web service only to those records that were modified after the last update. Communication with service will be realized via https, that is, SSL protocols so that it is necessary during the data exchange implementation to takeover the appropriate electronic certificate from the Agency. Authentication will be implemented in a way that username token is attached to a SOAP message header. Data exchange between the systems can also be realized via web service with clearly defined groups of data subject to exchange via XML schemes and also implemented AAA (Authentication Authorization-Audit) procedure up to the level of the individual data records.
Proposed solution will implement automatic, periodical synchronization with BRA agency.

Reference Data Module

Reference data module enables the creation and maintenance of controlled vocabularies (codebooks) for use in dropdowns and controlled fields within the various forms. The use of this module is restricted to the Reference Data Manager, the person responsible for maintaining the codebooks. It allows for the creation, update and amendment of reference data.

Information system presumes existence of several codebooks:

- Organizational Structure Definition - Organizational structure is a system used to define a hierarchy within an organization. It identifies each job, its function and where it reports to within the organization.
- Business Activity Definition – Adequate business activity is to be chosen upon registration process of FBO, among values stored in the codebook.
- Food Type Definition – During registration process of FBO user marks one or more types, between values specific to business activity.
- Retail Shop Type Definition – Implies values such as: selling packed and unpacked products which require refrigeration/freezing, selling packed and unpacked products which do not require refrigeration/freezing, selling only packed ambient stable products, with production of ready-to-eat food in the facility.
- Activities Duration Definition – FBOs are able to perform seasonal or all year around activities.
- Risk Criteria Definition - Risk criteria are terms of reference and are used to evaluate the significance or importance of organization's risks. They are used to determine whether a specified level of risk is acceptable or tolerable.

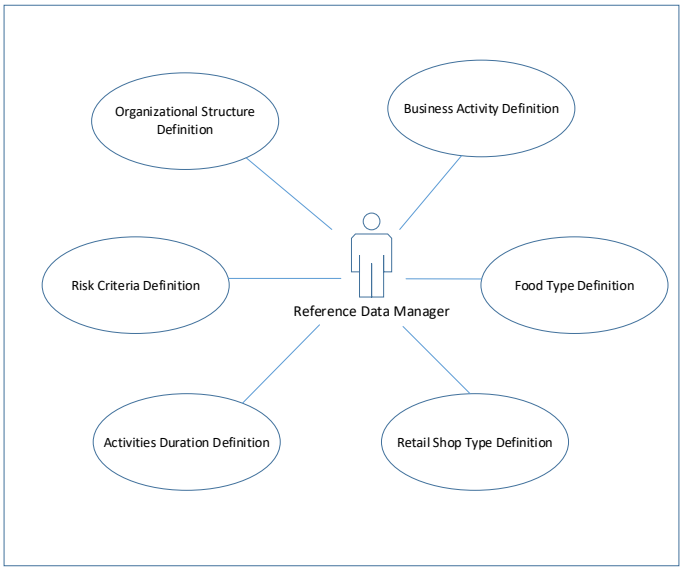


Figure 35: Reference Data Module functionalities

Software Architecture

The software architecture is built up over defined Conceptual architecture. Reusable components of Software architecture are used according to results given by Logical architecture. At this point there should be all the prerequisites for the implementation of behavior and structure of the FBOS system, since the non-functional requirements are already taken into consideration and infrastructure layer is defined. Existing software components are running by interpreting system specification or through generated source code. Figure 15 shows the organization of software components grouped into the logical spaces.

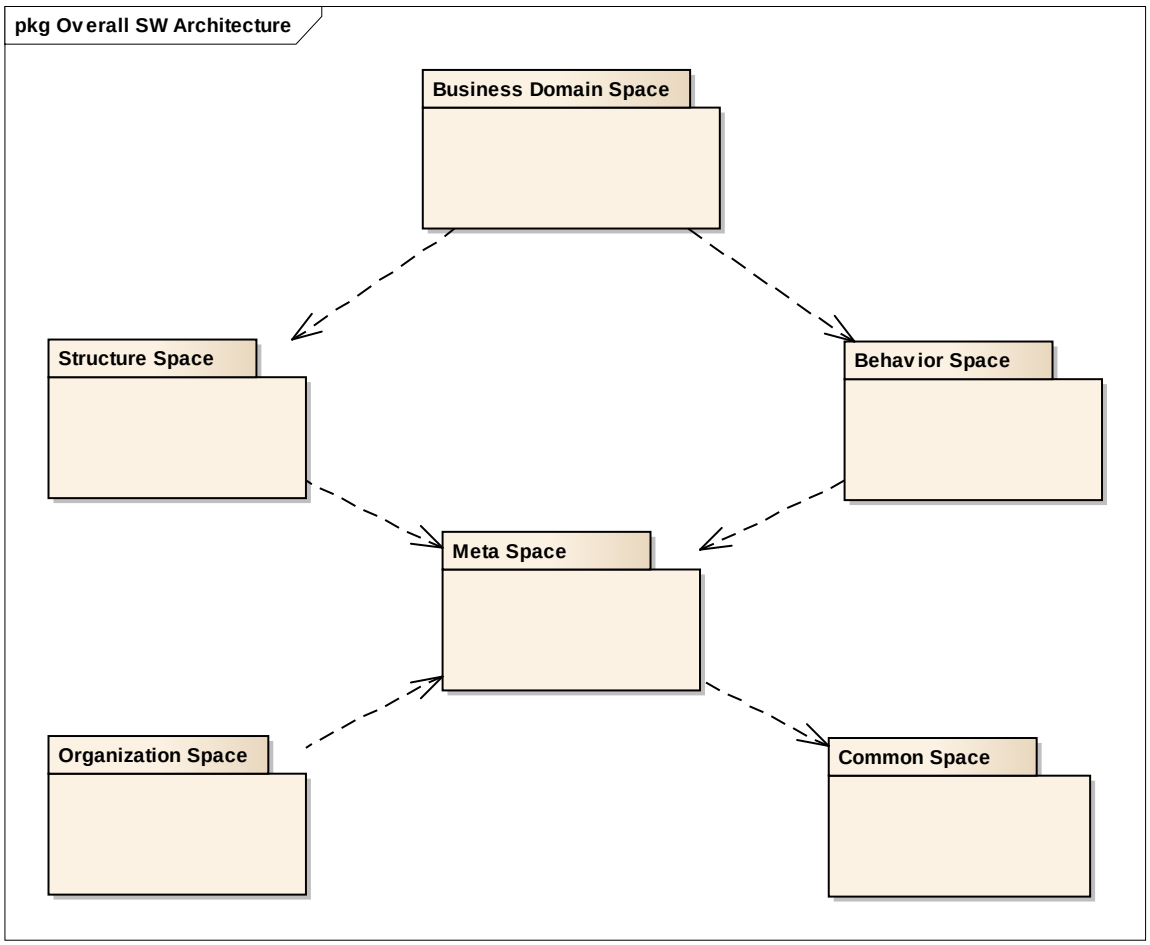


Figure 36 Overall Software Architecture

Target MDA architecture is implemented through the Meta Space, where meta-models that describe the whole system in general are stored. Software architecture is a result of the Construction phase.

FBOS will be developed as web-centric multilayer architecture based on Service Oriented Architecture – SOA using SOAP protocol for integration with extern systems and using REST architecture for integration between internal components.

Presentation Layer designated for end users according to FBOS user needs will be developed using following technologies:

- Web application based on HTML5 standard developed in ReactJS environment
- SecurityTray component designed for Client qualified certificate authentication

Business Logic layer consists of several middleware components and will be developed and operated on Microsoft technologies, specifically:

- FBOS Core Business Logice component based on DDF framework;
- Security Server component designed to validate qualified certificate authentication, to timestamp and to archive signed documents;
- FBOS Service Component designed to integrate with external systems;

Data Layer will be based on standard Microsoft relational database management system:

- SQL Server and its advanced features
- MS Integration Services

All sub-chapters and figures above in this chapter have purpose to describe maturity of used development process and runtime environment for proposed solution of FBOS system and its core components. The key advantage of this approach is the fact, that it is used for many years in real production environment on many successful projects.

Re-use of all technologies will allow us to focus primarily on business functionality and will provide proven, stable and scalable system developed in predicted time and quality.

Business Domain Space

The Food Business Operator System will provide easy to use Food Business Operator and Food Facilities Register, along with support of Inspection planning and oversight.

The proposed solution will have the following segments:

- Segment open to the inspection only
- Segment open to the inspection and
- Segment open to the general public.

The register will communicate and exchange data with Business Register Agency in order to integrate data of interest for Food Business Operators and Food Facilities.

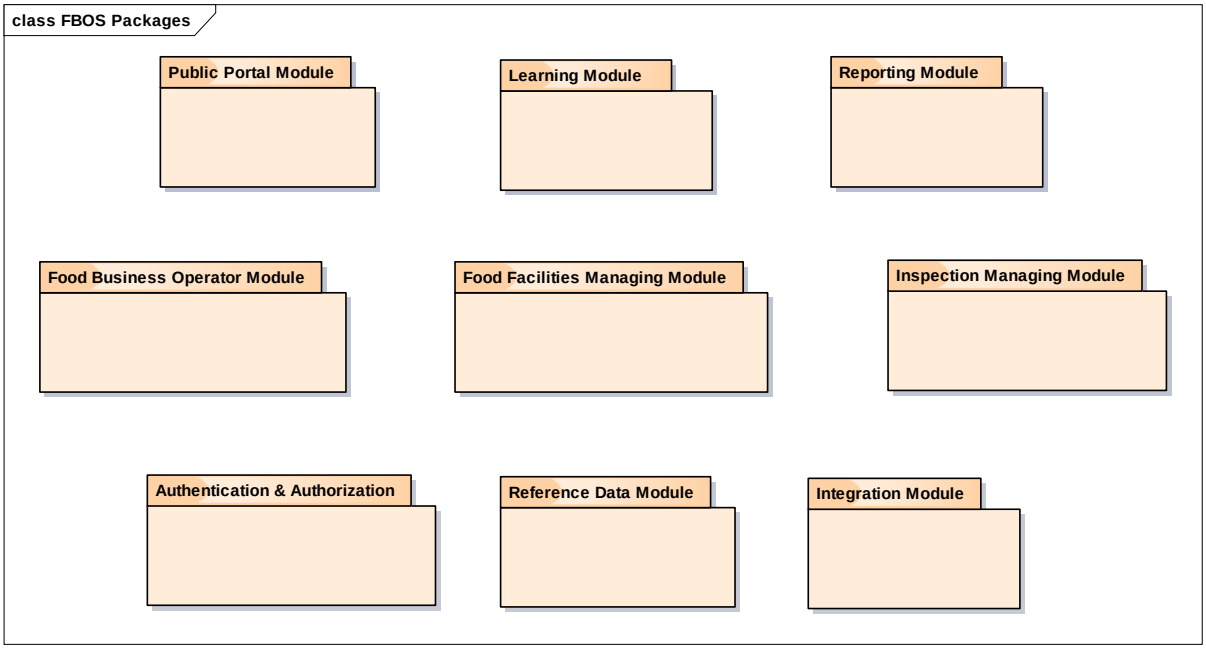


Figure 37 Business Domain Space

Business Domain Space e.g. FBOS system will be built on top of the software components described in following sections.

Meta Space

Meta Space includes software components responsible to provide MDA environment. Belonging software components are:

- Information System Repository Component which specifies system business and technological domains.
- Administration Component is application which supports model management and formal specification of system structure and behavior.

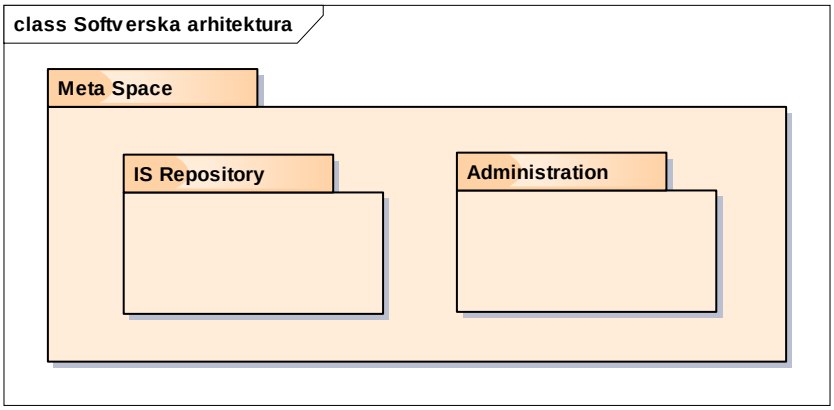


Figure 38 Meta Space

IS Repository

FBOS will be based on Information System Repository, that is, on information system which specifies system business and technological domains. IS Repository is also supposed to be adaptive to frequent functional, organizational and technological changes.

IS Repository has twofold role:

- “passive” when it is used for specification of domain concepts, and
- “active” when it is used for dictionary driven applications whose execution is based on domain concepts specification. Those applications are more “interpreters”, rather than “hard coded”.

Some IS Repository models are specified at upper MDA levels as depicted in Figure 2. Based on these principles, IS Repository facilitates solving of permanent change problem and enables interoperability (which also implies code generation).

In this way, IS Repository becomes core DDF component for rapid IS development. Additionally, rapid development is supported by developing several general software components (persistency, transaction management, access control policies, error handling etc.) which provide infrastructure layer as presented in Figure 15.

As discussed above, DDF becomes an integral part of software products, with major features: formal system specification and predefined software architecture.

Another important feature of DDF is tool for automated model transformation and code generation. In our approach, we consider software development process as set of consecutive model transformations up to the final executable model on target platform. Resulting IS is comprised of related concrete models which conform to potentially different meta-models (relational, object-oriented, XML Schema, OWL etc), as well as models specified in initial phases of software development process (SSA, UML, E/R, BPEL etc).

This approach includes MDA principles, that is, it separates the specification of functionality from the specification of implementation of that functionality on specific technological platform.

Administration

Administration is an integral part of IS Repository in order to support model management and business system specification. Administration is compound of several applications/modules:

1. Structure Module – describes the structure of the domain concepts in accordance with the Domain Model Description. DDM is in itself recursively described as a meta meta model, and is used to specify other metamodels.

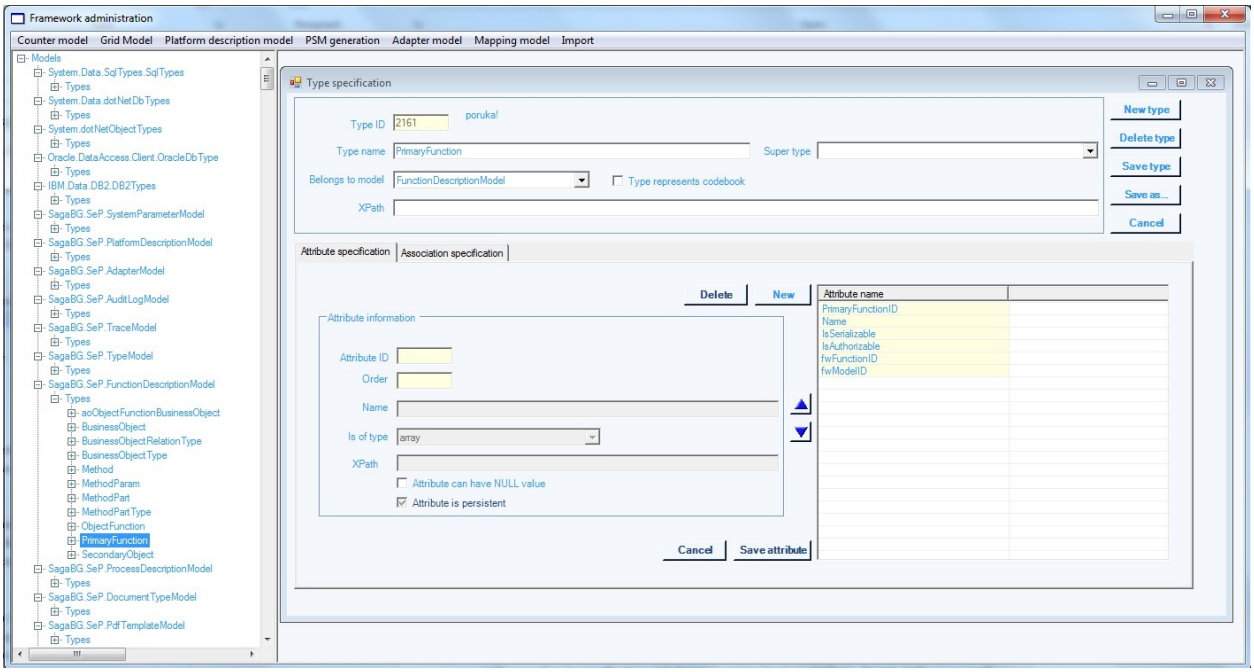


Figure 39 Structure Module

2. Business Process Module - provides formal specification and business process automation. Compliant with WfMC standard.

3. Function Module - specifies the business logic and system behavior of some particular business functionality. Based on an improved SSA method.

4. Adapter Module - used to work with a variety of technology platforms. For now, it is adapted to work with: FileNet, BizTalk, SQL Server, Oracle, DB2, Primavera, XmlSchema, MS CRM, SOAP technologies

5. Organizational Structure Module – can include different organizational models: matrix, functional, project and other organizations.

6. Document Module – defines the different types of structured and semi-structured user documents.

7 Security Module - manages user access rights. Authentication and Authorization may be realized using several different security policies (username/pasword, PKI, Smart Cards, Tokens ...)

8 Notification Module - integrated with Scheduling model, provides information based on events or time in the form of e-mail, SMS or events triggered to other modules.

9 Mapping Module - used for the transformation of the model on the technological or business level. It is prepared to execute upon XML, relational or object model (Java, C #) as source and target models. In addition, it allows the transformation of specific business models (e.g. B2B transformations)

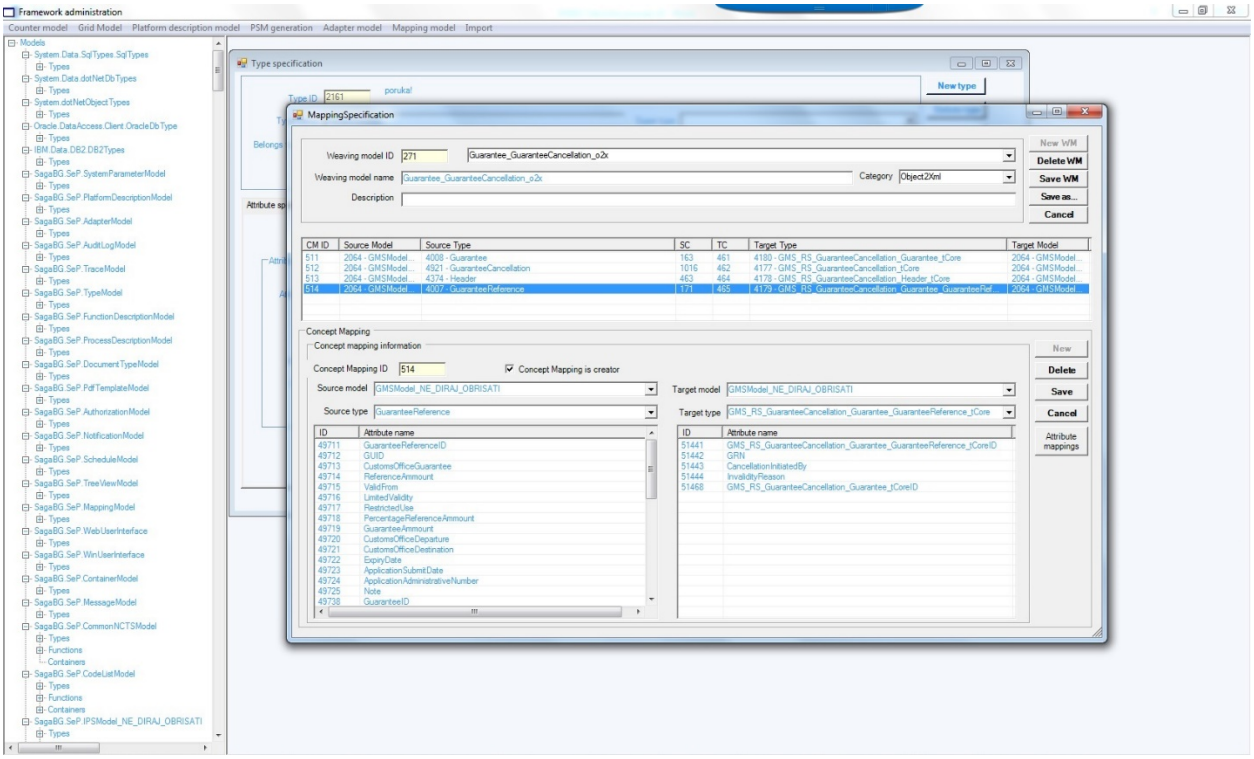


Figure 40 Mapping Module

10 Audit Log - records any kind of change in domain objects (create, modify, delete), who and when changed data and which function was used in that operation.

11. Code Generator - significantly speeds development and reduces test resources by generating source code according to specification and chosen target technology platform. Target platform is specified in so called Platform Description Model.

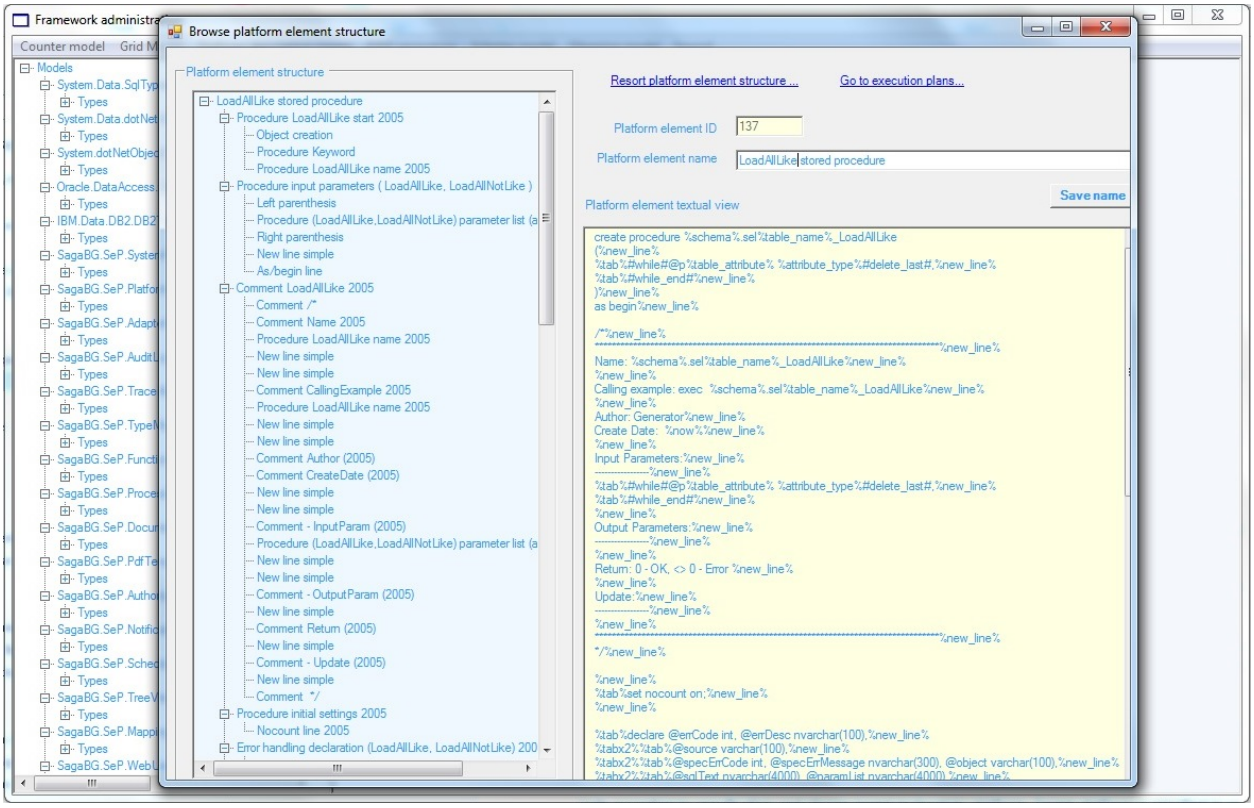


Figure 41 Code Generator

12. Deploy generator - generates a SOA environment with the appropriate proxy and façade layers.

Behavior Space

Behavior Space consist of software components which provide realization of system behavior. Included software component of Behavior Space are:

- Business Processes Component enables formal definition, control and automatic execution of complex business processes upon workflow engine.
- Business Functions Component enables formal definition and execution of essential business functions which are performed over organization resources.
- State Transition Component enables managing of organization resource lifecycle
- Business Rules Component provides formal rule definition and rule execution upon business rule engine.

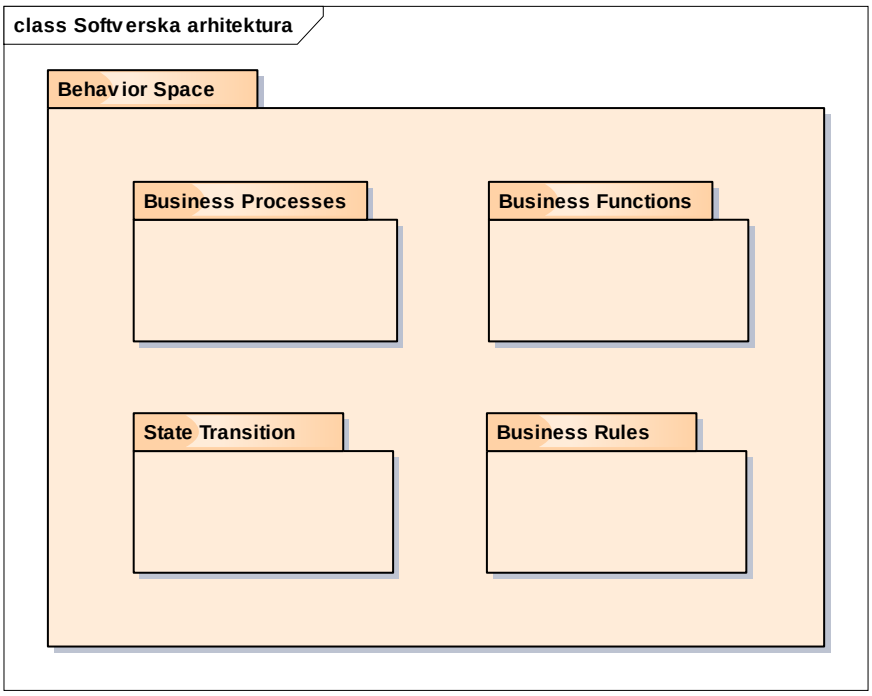


Figure 42 Behavior Space

Business Processes

Business Processes Component enables automatic process execution driven by definitions specified in Business Process Module (Administration application from Meta Space).

Business Process Definition consists of activities that represent the basic building blocks of the processes. The activity represents an atomic unit of work, which means that it is executed completely or not executed at all. Activities are connected with other parts of the system so that the activities represent separate business functions which are described in Behavior Space (Business Functions Component). Also, the activities are performed by the perpetrators, who have the appropriate role (ie. The right to perform a certain activity) in the process, manually or through an application, with the added qualitative value to the resources over which the process is running.

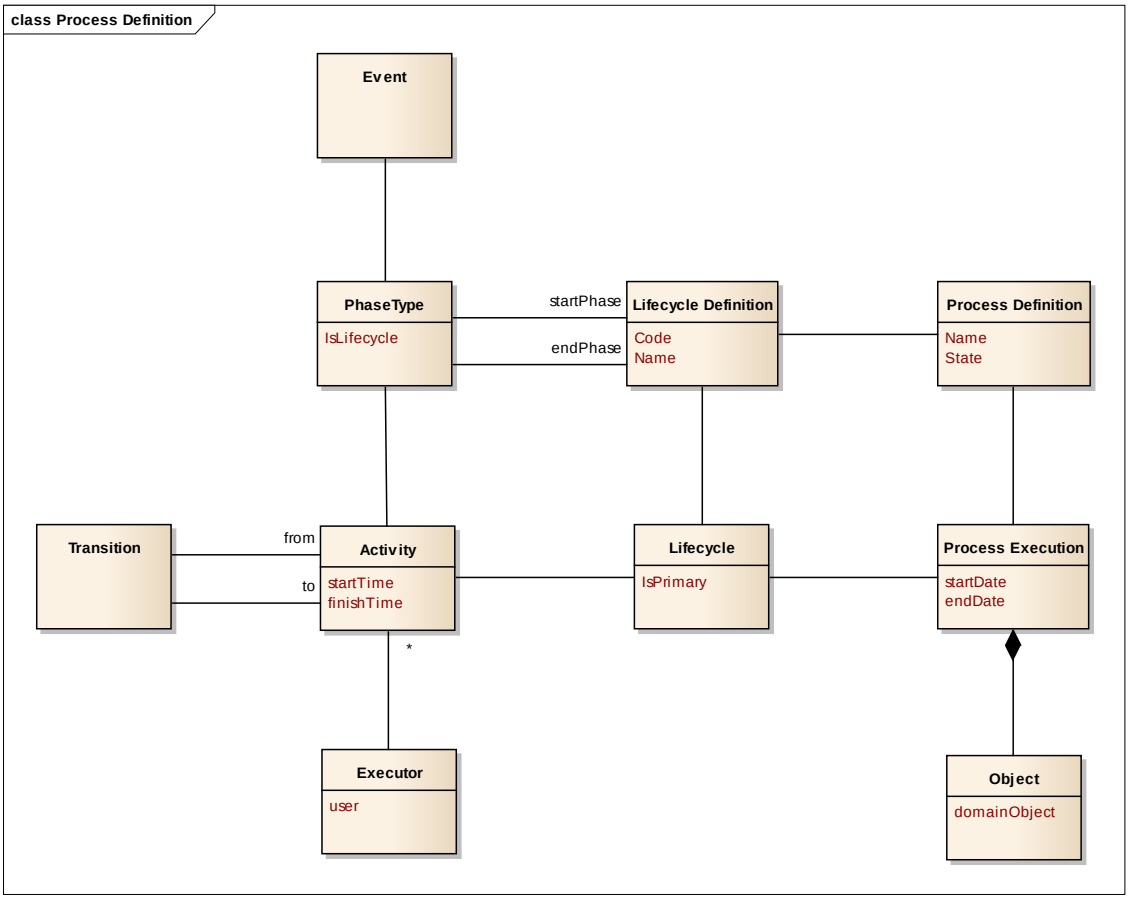


Figure 43 Business Process Definition

As shown in Figure 22, when describing business processes a start point is the fact that during the process running some objects' values are being changed. Value changes are occurring by precisely defined sequence and by the precise conditions whose specification represents the description of the given object's behavior or its life cycle. Through various processes single object can pass through the different life cycles.

The concept of **lifecycle** is described as a set of possible phases through which the object passes during the work flow and the conditions of transition from one phase to another. The important thing is the fact that the phases of a lifecycle are being performed sequentially, ie. only after current phase is completely finished, it is possible to start the next phase.

We may conclude that **phase** represents an indivisible entirety which can be in an *active state*, when one or more business activities are being performed and the state of object is being changed, or it can be in a passive state, when no activity is being performed. Transition to the next phase may require some condition fulfillment.

The next concept is the **event**, which is generated by stimulation from the environment or within the system (customers, managers work or executor of a task) or by completion of certain lifecycle phase. What is important is that events initiate the lifecycle, so several parallel lifecycles (and their phases) could be performed at the same time, where events translate lifecycle to the next phase.

Moving from one phase of a lifecycle to another is called **transition.** If we look at lifecycle as a graph, then the nodes of the graph are phases and arcs which connect the nodes are transitions. The event triggers the transition, but it may be limited by the additional condition..

From one phase there may be multiple transitions, whereby transition conditions and events are mutually exclusive, so it is not possible to start parallel phases within one life cycle. this way, the alternatives are modeled.

Dynamic of the business process is described by defining the objects and their lifecycles. Full job description requires the specification of the resources needed for its performance.

The concept of **business process**, contains the basic lifecycle that can be run by one or more other, parallel lifecycles. The business process consists of a sequence of activities that take place in certain stages of lifecycles initiated.

Business Functions

As shown in Figure 23, system functions are formally described in the metamodel *Function Description Metamodel*. The basic concept of the model is the Primary Function, which is performed over one or more components of the system (Business Object), while considered function's interface could be formally defined (via the Method concept).

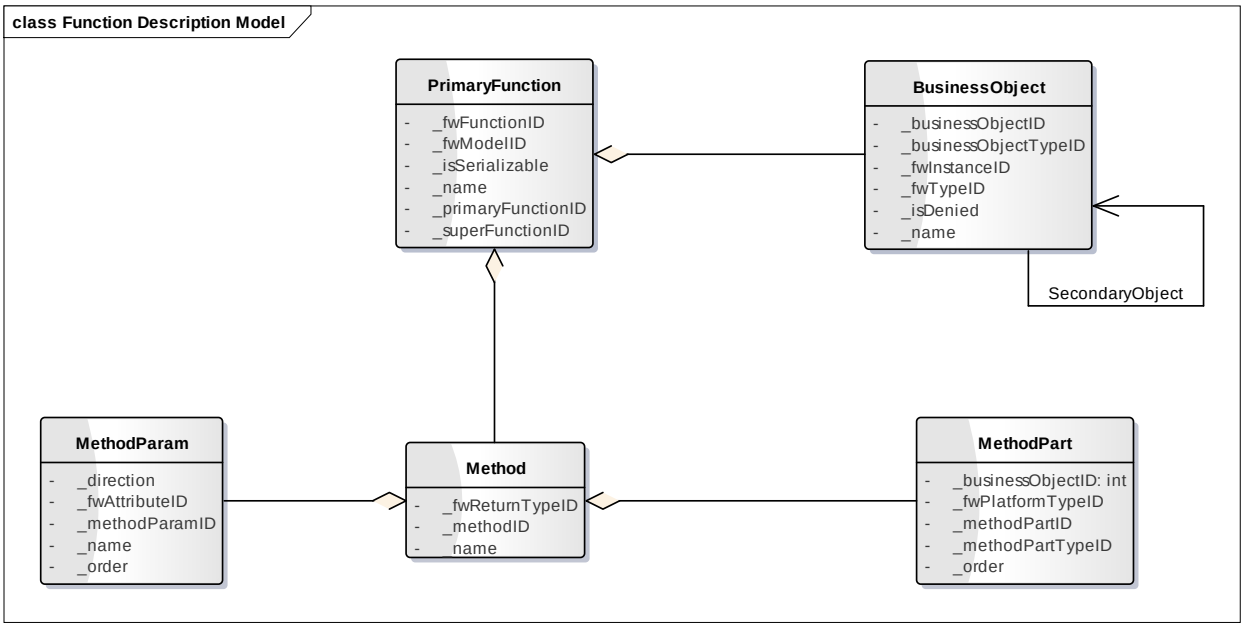


Figure 44 Object - oriented model of Function Description Model

All depicted functions are implemented as a Function object. This way a single function of the system is implemented as a class that implements the behavior (logic) of the related system function. Since the system function is performed over the corresponding system objects, each implemented function has access to the system entities through a set of predefined CRUD and other related operations.

This means that any external user - system communication, as well as download or changing of a system components can be made only through corresponding system function.

This system functions design also enables a unique way of checking user authorization.

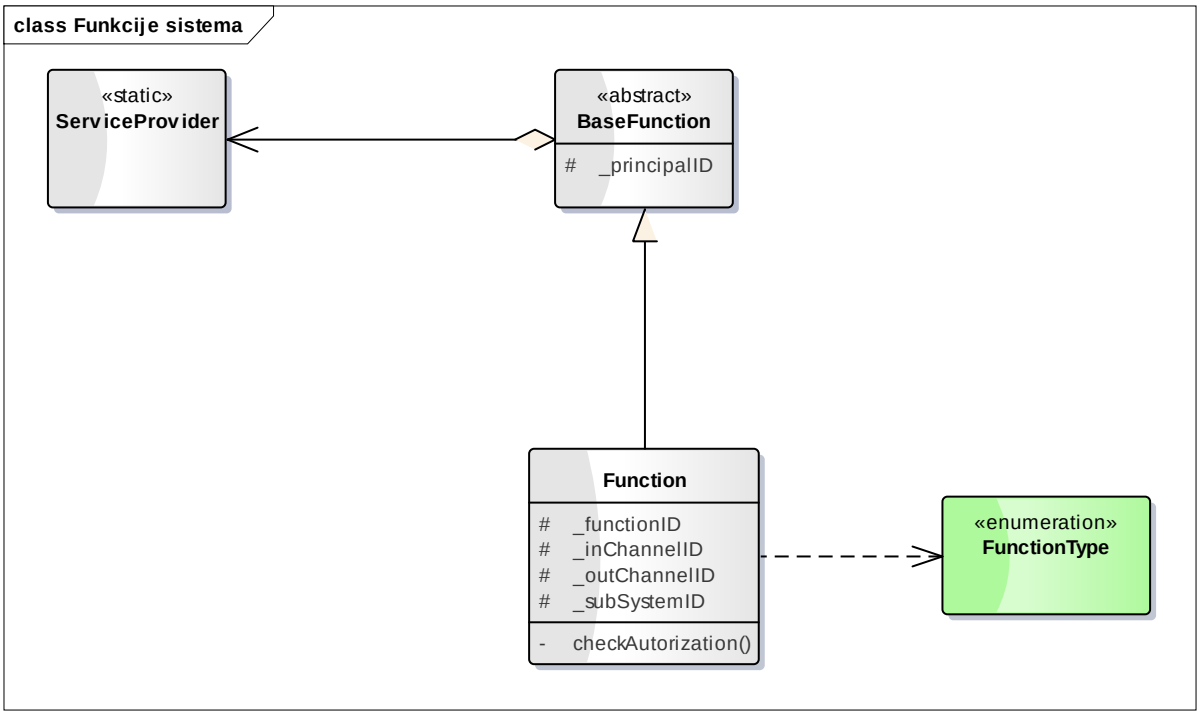


Figure 45 Object - oriented model of the System function component

State Transition Component

State Transition Component is used to perform and manage state change to objects which may be in different states during their lifecycle. Object is transitioned from one state to another when matching **event** is triggered. While object is in current state (or when object reach or leave some state) appropriate **action** may be performed. Actions may be new events or business functions.

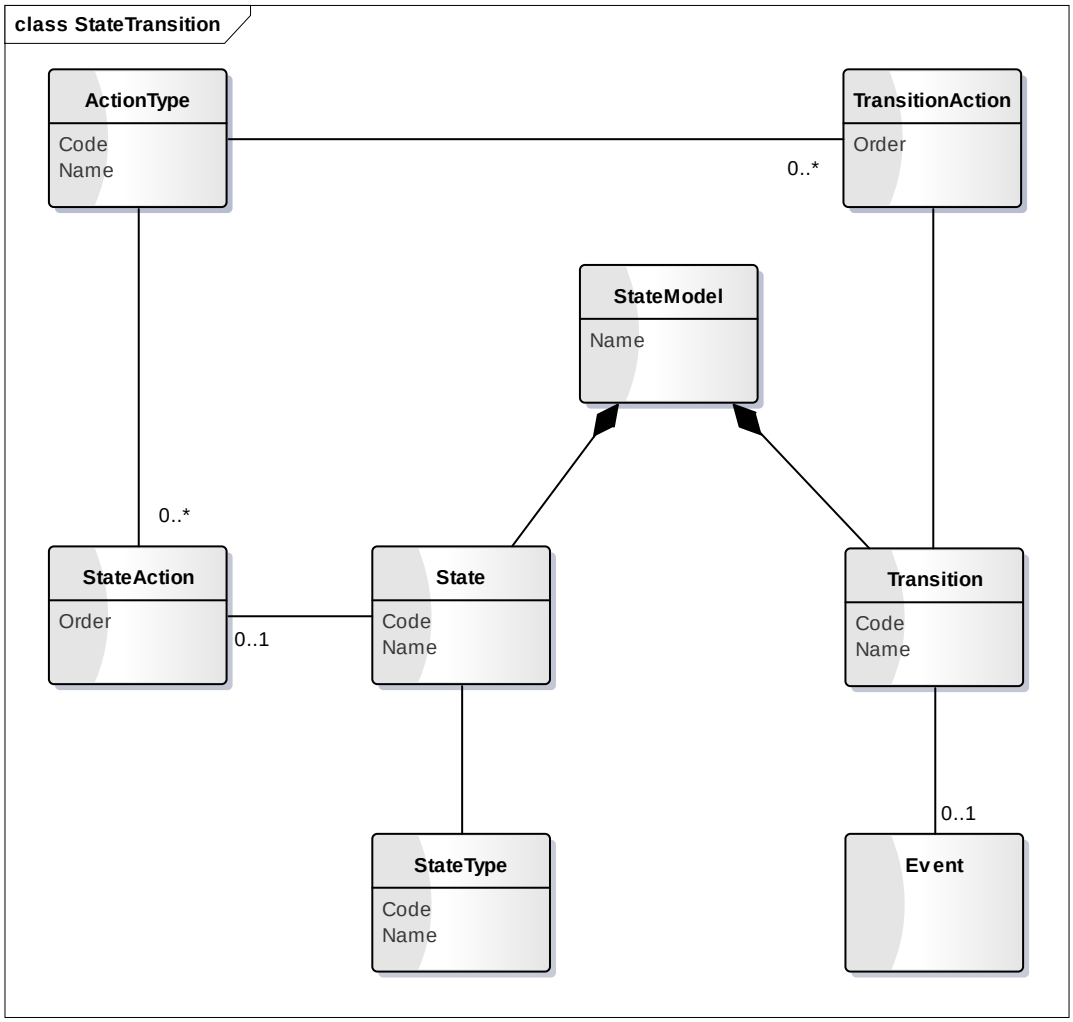


Figure 46 State Transition

Business Rules Component

Business Rules Component provides engine capable of linking formally defined rules to business objects, relationships, attributes, functions, processes and other concepts of interest.

Each rule is compound of expressions, which consist of left and right operator and operand. More complex expressions are built up on simple expressions thus making complete rule as depicted on Figure 26.

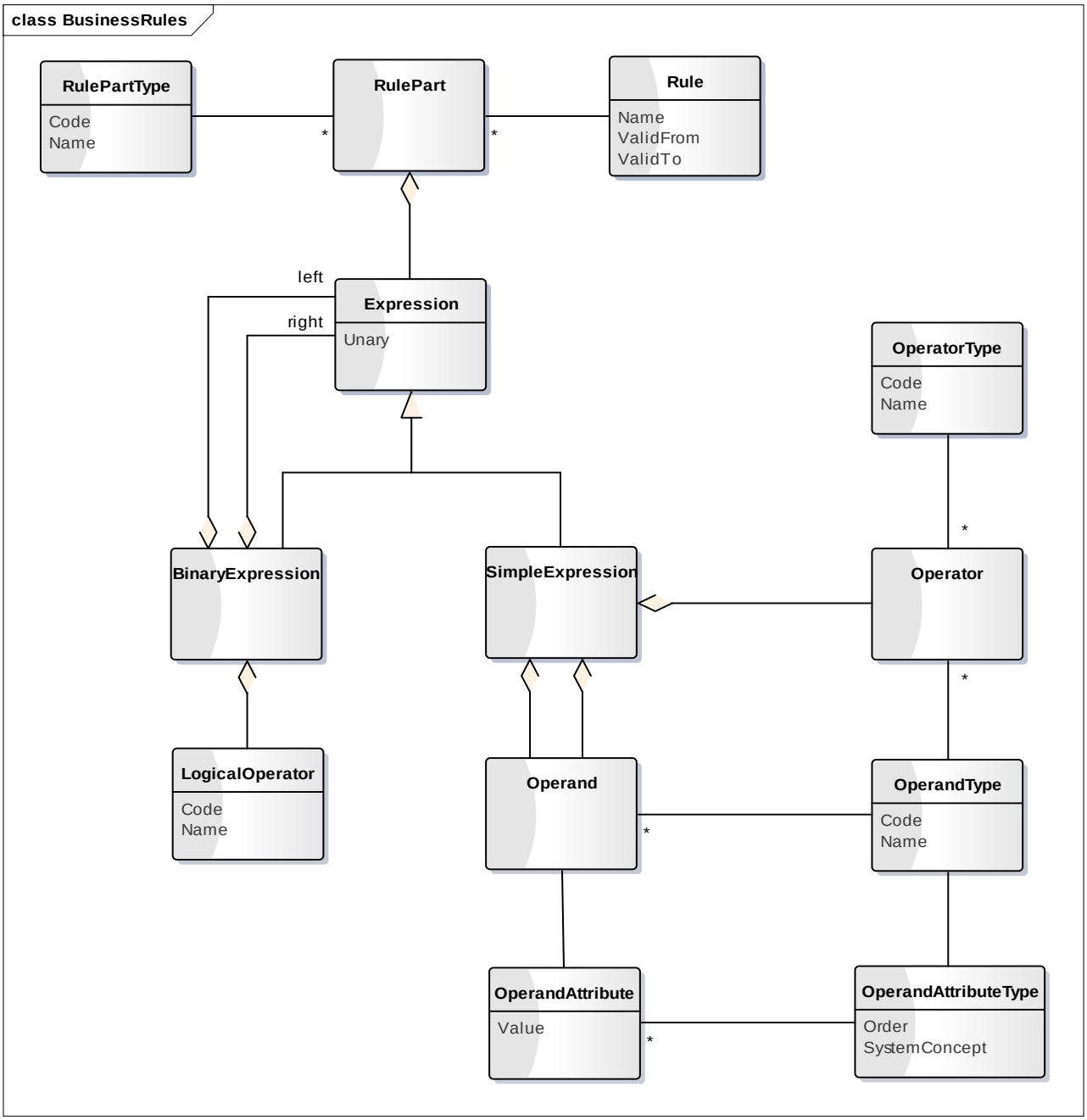


Figure 47 Business Rules

Organization Space

Organization Space is compound of software components responsible to define and drive organizational structure, employees and access policies.

Belonging components of Organization Space are:

- Organizational Structure Component enables specification of concrete organizational structure
- Authorization Component defines access policies and manages concrete access rights.

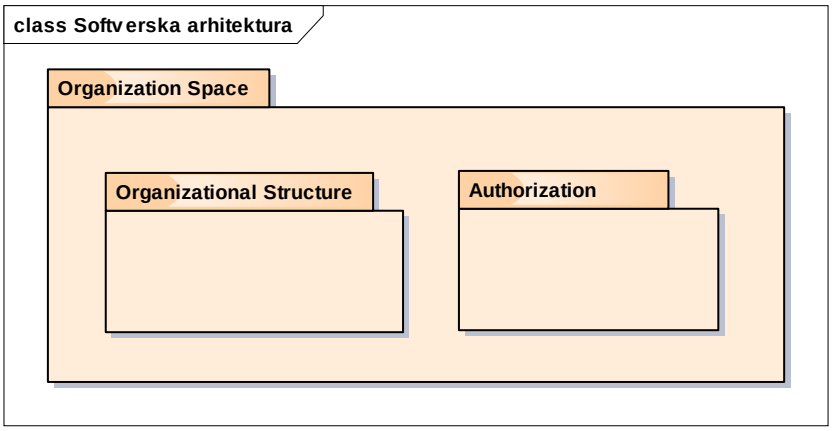


Figure 48 Organization Space

Organizational Structure Component

The organizational structure is used to record information about the organization and its staff. In addition to basic information about organizational units and employees, this component connects system users with access to appropriate facilities and the performance of the system functions via Authorization Component (from Space Organization).

In Figure 27 is presented model of organizational structure. The basic concept of the model is **OrganizationalPart** and it represents all the organizational units. As a separate subtypes of this concept are defined **Position**, **OrganizationalUnit** and **Organization**.

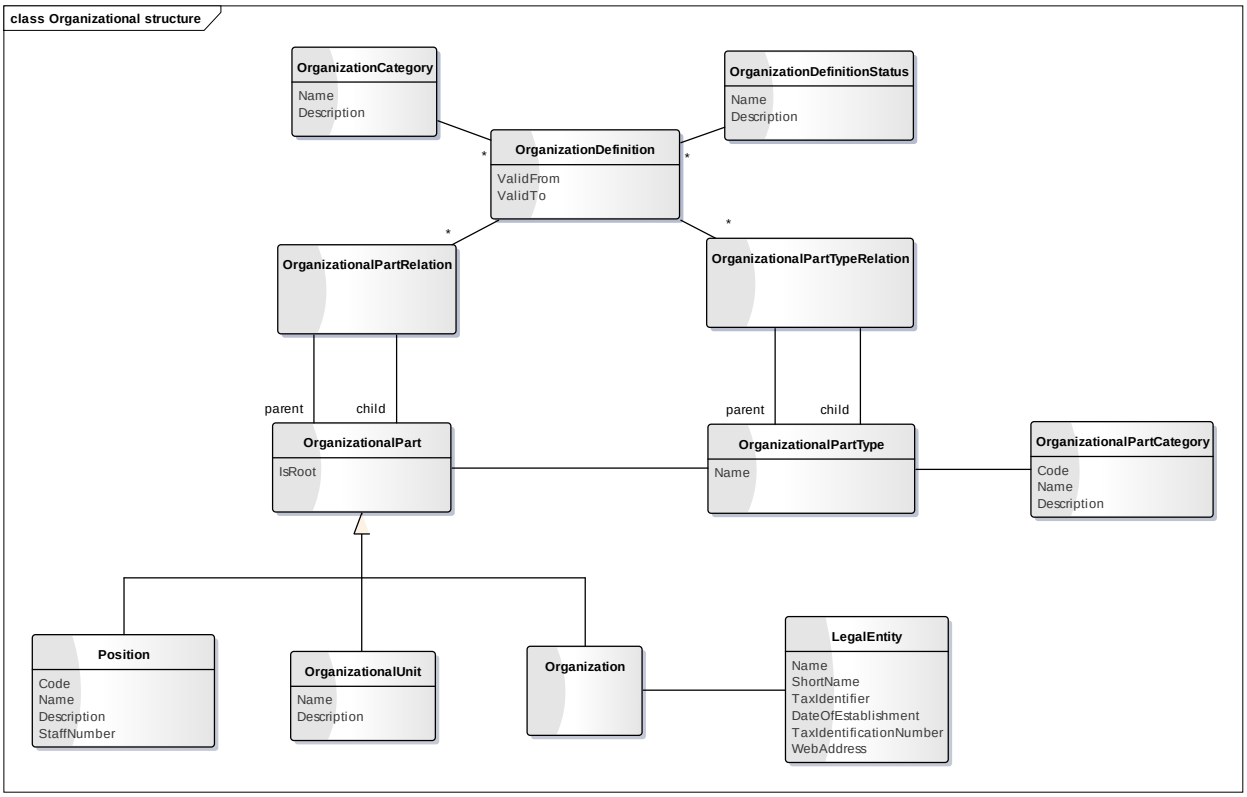


Figure 49 Organizational structure

Position is a set of all units on functional or some other criteria, whereby it is common for a worker in the structure can be solely related to workplace. The **OrganizationalUnit** concept represents all the remaining parts of the organization, whose type (eg, Division, Department, etc.), due to the flexibility, belongs to **OrganizationalPartType** in order to dynamically introduce a new kind of organizational unit. Type of the **OrganizationalUnit** is a concept that includes all existing types of organizational divisions within the organization.

Authorization Component

Authorization is software component which manages access policies in FBOS. Authorization is a system function which has two aspects:

- Definition of access policies which specifies mappings of roles with business functions and objects.
- Enforcement of approving or disapproving access requests.

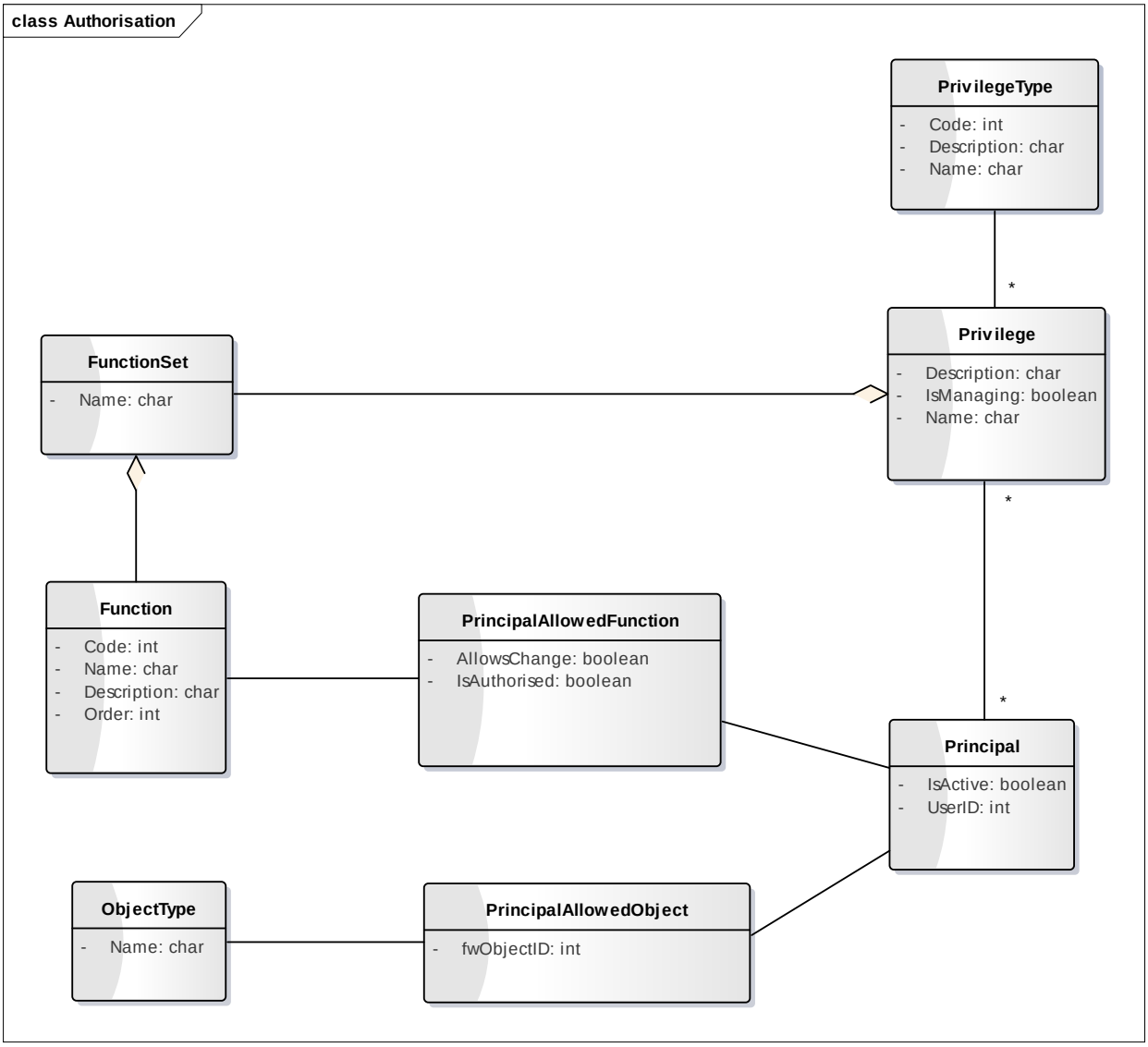


Figure 50 Authorization

As shown in the figure, each system user that can be Legal Entity or Individual (presented by Principal concept) has the appropriate Privilege for system use. Each Privilege is of a type (**PrivilegeType** which may be *Identity*, *Group* or *Role*). This way specific users can join adequate role or group. The privilege may refer to the appropriate set of system functions (**FunctionSet**), whereby the user who has the privilege for a particular set can perform all functions of this set. A set of functions is a set of at least one system function (**Function**), which representsis functions identified in the functional model. Each function during its execution is potentially performed over system resources, which is shown through a link with the **ObjectType** concept.

Privileges can also be refered to a particular system object, so that at checking access rights it is controlled whether the user has access or modification rights to a single object.

Structure Space

Structure Space includes software components responsible to manage structural features of business system. Belonging Structure Space components are:

- Service Layer Component is interface layer to system structures and holder of cached collections used by other software components.
- Transaction Manager Component is responsible to manage connections and transactions in proper way.
- Identifiers Component has responsibility to provide uniqueness of domain objects and other concepts when needed.
- Domain Objects Component provides business entities specification and implementation.
- Persistance Layer Component manages object – relational mapping.

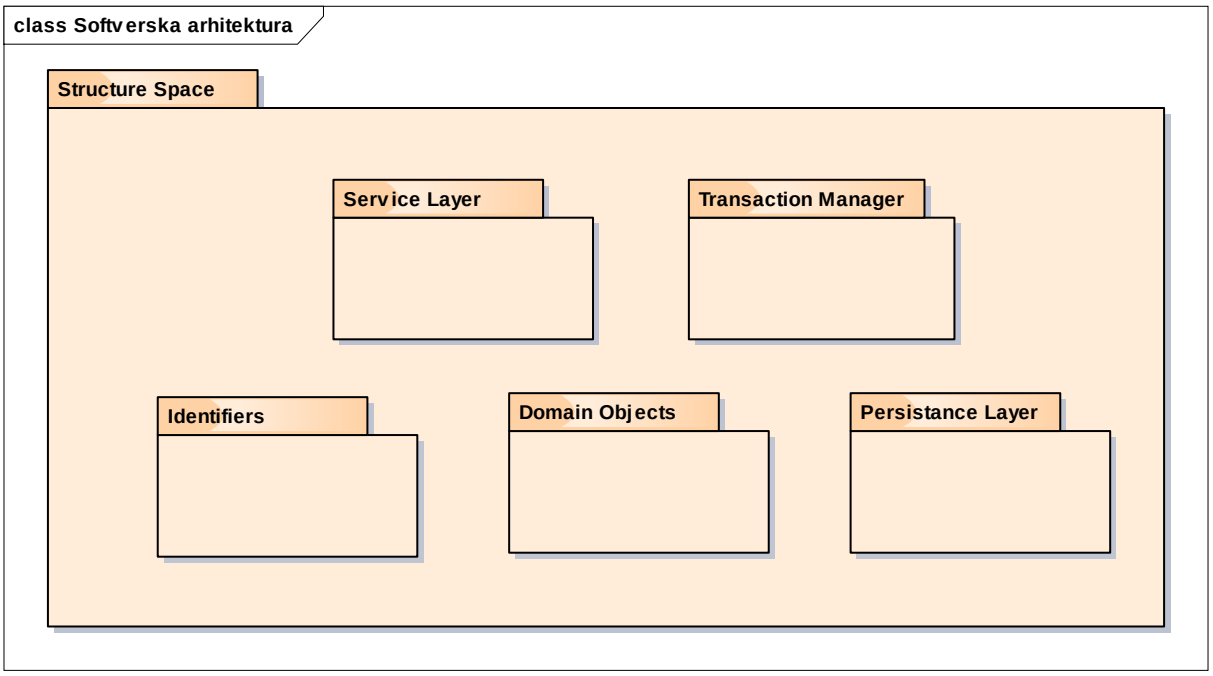


Figure 51 Structure Space

Transaction Manager

Transaction Manager is component responsible for connections and transactions managing. This way consistent state in the data model is provided as well as the timely opening and closing connection to the data layer.

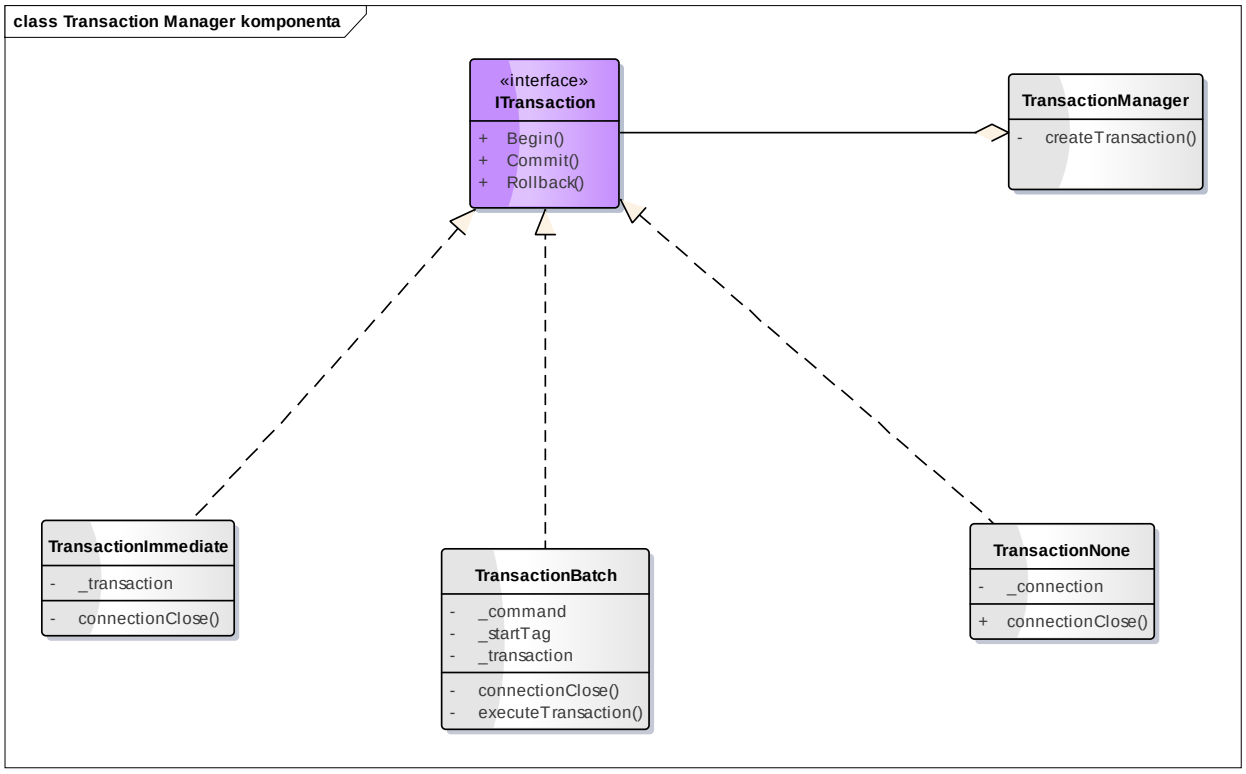


Figure 52 Object - oriented model of Transaction Manager component

It is defined more types of transactions:

- *BatchTransaction* – on the framework level beginning of the transaction is emulated, and after saving all the commands that belong to the transaction, it will be executed in a single call to the database. Thus defined logical transaction enables faster performance and reduce the load of DBMS. Also, in this type of transaction calls, 'malicious code' protection mechanisms are built in.
- *ImmediateTransaction* – transaction that starts and ends at the middle tier using a standard ADO transactions.
- *Transactionless* – when a call to the database contains single DML command, whereby the transaction does not start

The way shown, user (programmer) never explicitly starts (nor closes) connection or transaction. This way the system is insured against accidental error or from the unnecessary burden of DBMS.

Identifiers

Identifier manager is responsible for all the entities of the same type in the system to have unique identifiers. This will ensure uniqueness of entities in the system. Uniqueness is provided by reserving certain amount of identifiers which are predefined. This way it can be reserved a greater amount for often created objects. It is also possible to determine which server will reserve the identifiers for which types of entities.

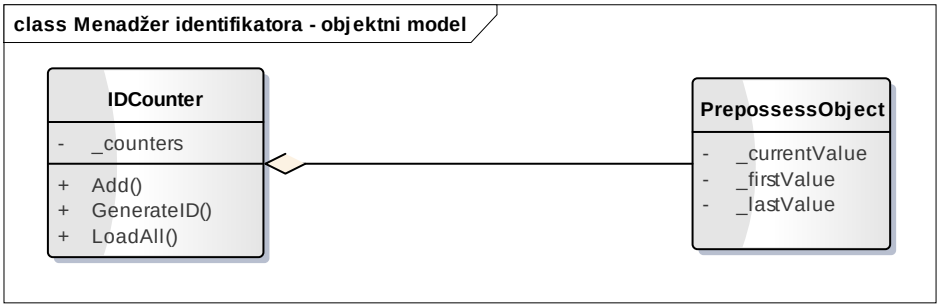


Figure 53 Identifier manager – Entity model

In order to speed up the assignment of unique identifiers, reserved quantity is cached in the model shown (Figure 32), and is assigned upon request.

Service Layer

Service Layer Component is based on singleton object (Service Provider) which holds all cached collections needed to system proper running. This component also provides interface to upper system components, such as Business Process and Function Component (from Behavior Space). As central component of Structure Space, Service Layer Component integrates different parts of infrastracture layer.

Integrated components are:

- Domain Objects
- Identifiers
- Transaction Manager
- Persistance Layer
- Audit Log
- Trace Subsystem
- Adapters

Cached collections, loaded from meta database, hold by this component are:

- ObjectModelColl
- FunctionModelColl
- WeavingModelColl (e.g. Mapping)
- StateModelColl
- BusinessRuleColl
- IdentifiersColl
- AdapterInterfaceColl
- AuthorizationColl
- SystemConfigParemetersColl etc.

Domain Objects

Domain objects represent business entities which specification is defined in DDM model. Specifications are loaded and cached at system startup. This way, loading entity's description from a data source (database, xml document etc.) does not slow down the system.

DDM model as depicted on Figure 33, (model on the M2 level), is a model that describes the system structure (entity types, belonging structure and their relationships).

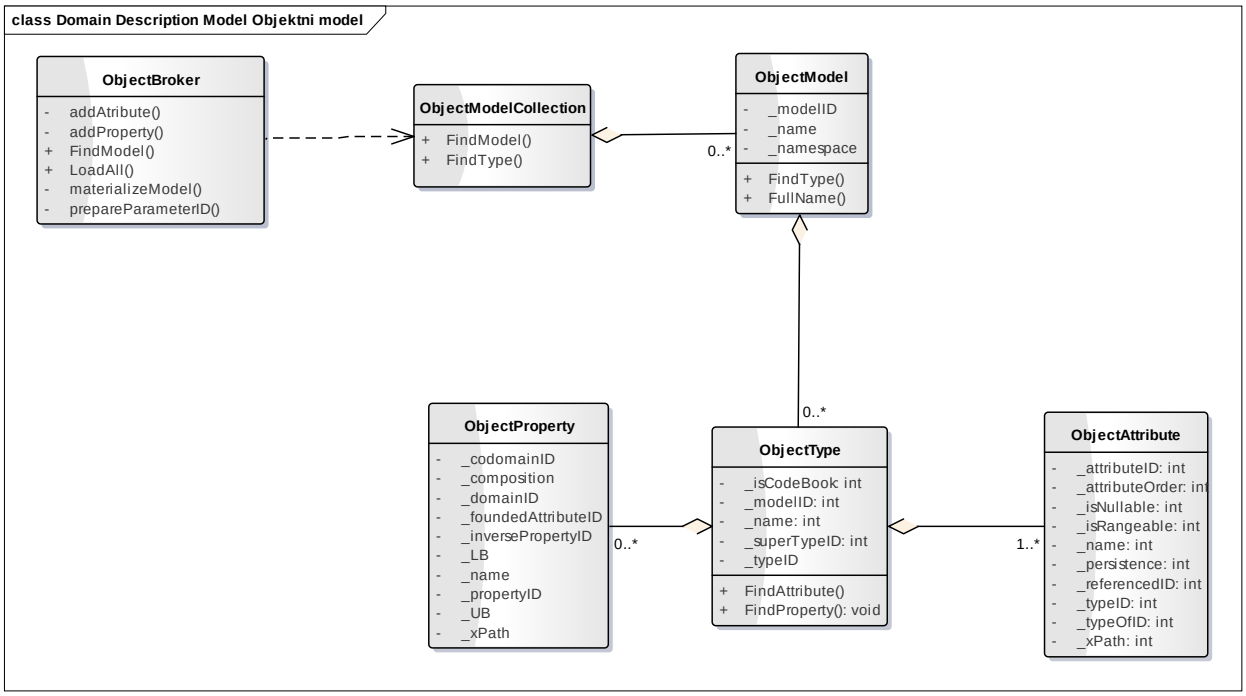


Figure 54 Domain Description Model

An entity can be associated with other entities through a recursive relationship *association* when it carries information about the type and identifiers of entities related to it. Another defined connection is a ‘weak entity‘ role (connection *composite*) when an entity carries overall information about the weak entity. All domain entities are generic, their structure and relationships are read from the description stored in the dictionary. During the application runtime, *Domain Description Model* package takes charges for those definitions.

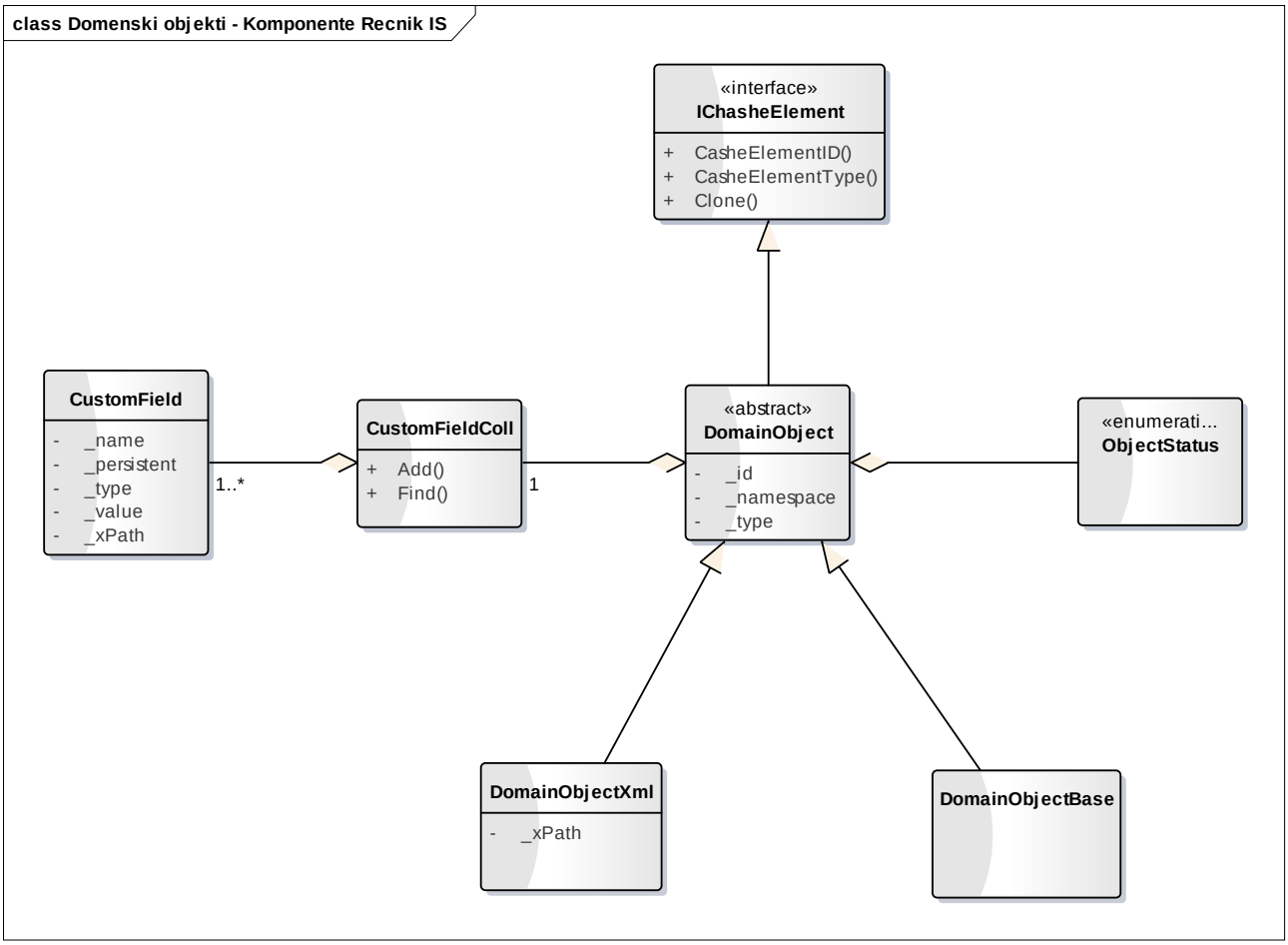


Figure 55 Domain Object

Persistance Layer

Persistance layer provides the connection of two different models within the system, the object - oriented and relation model (from the point of a three-tier architecture, this layer is connecting the domain logic tier and the data tier). The basic functions of this layer are materialization and dematerialization of entities, ie. conversion of data from relational database to an entity and vice versa, converting entity to relational model.

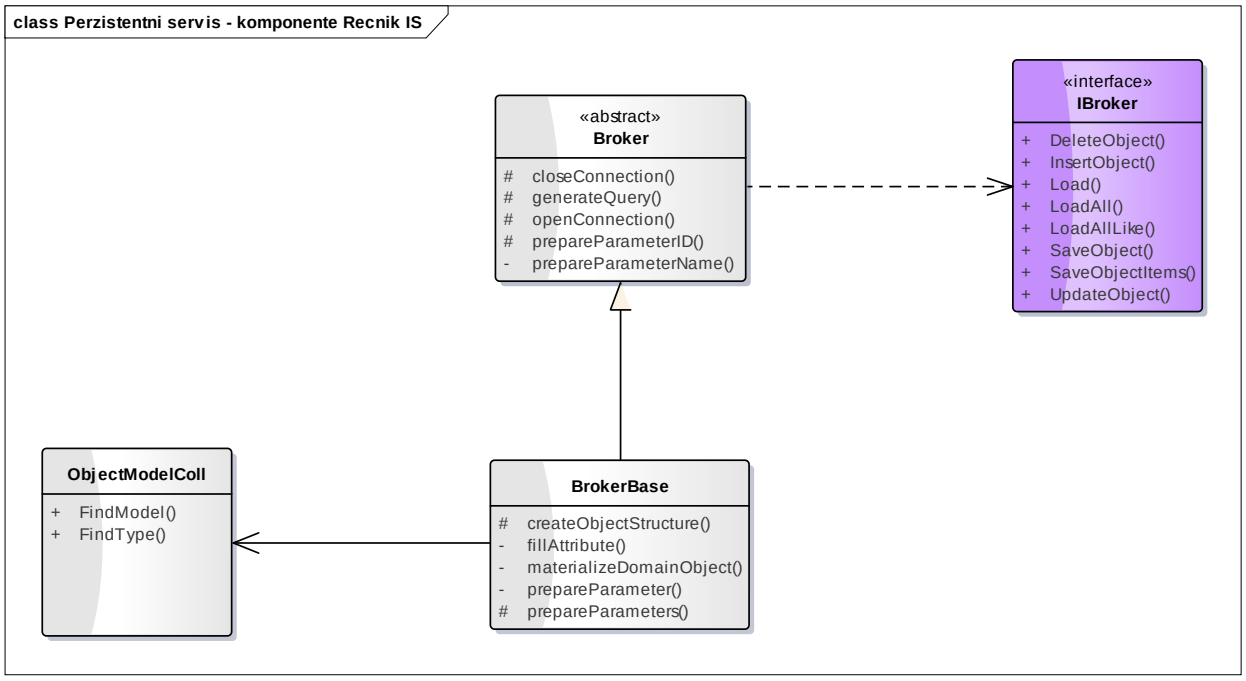


Figure 56 Persistance layer - Brokers

Persistance layer (Figure 35) shows that in addition to the materialization and dematerialization of objects, this layer has a predefined set of operations over the objects (CRUD operations). Connecting the object and relational model is performed through an interface (set of procedures of a specific DBMS) whereby definition of the input parameters for the interface at run time is "read" from the object specification in Data dictionary.

This connection is extra secured (from potential errors) because the stored procedures are generated based on the object description which minimizes the possibility of human mistake.

Common Space

Common Space holds helper components used by other Spaces. Software component included in Common Space are:

- Notification Component is responsible to specify, manage and send various types of system notifications
- Mapping Manager Component is used to definition and execution of object transformation whether from business or technological domain.
- Adapters Component facilitates comunication to external systems and their data or messages.
- Audit Log Component is provides tracking business operation at the application leve, for both, objects and functions.
- Exceptions Component is responsible to uniformly handle system exceptions, as well as the time and place of their occurrence.

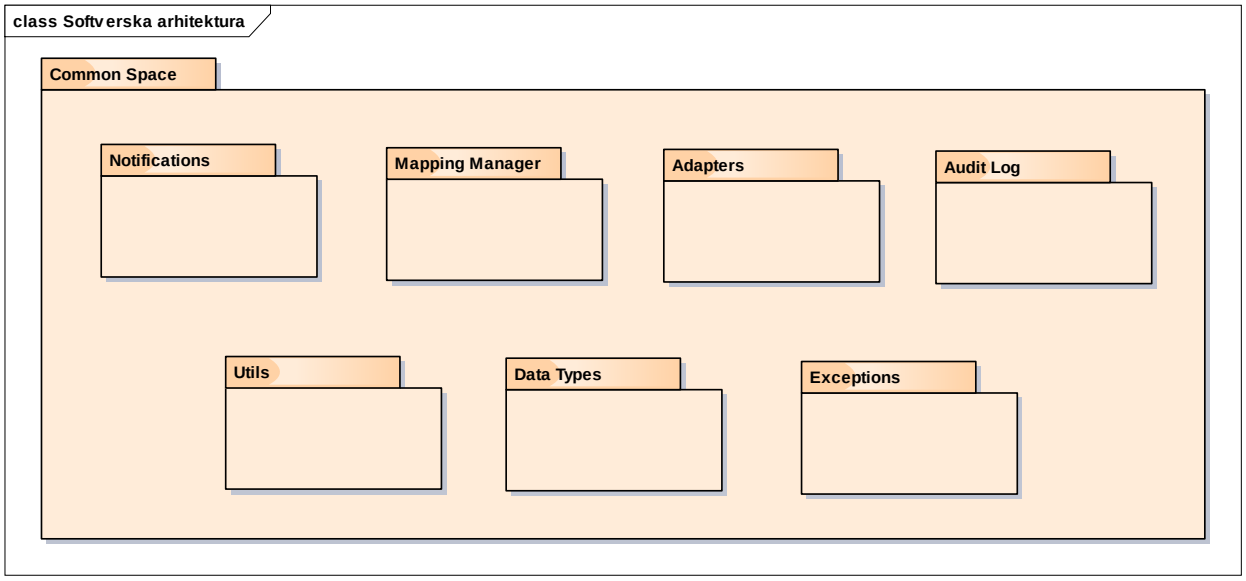


Figure 57 Common Space

Notifications Component

Notifications component is a set of functions which include work with different types of notification in order to monitor various events within the system or, on the other hand, system response by sending an appropriate notification in a predetermined period or on explicit user request. Notifications component allows to define, create, download, process and send various types of notifications, from multiple and to multiple different channels, towards multiple different users. It also provides system performance monitoring.

The subsystem can have its own or external verification of the user's authority to accept or require different notifications.

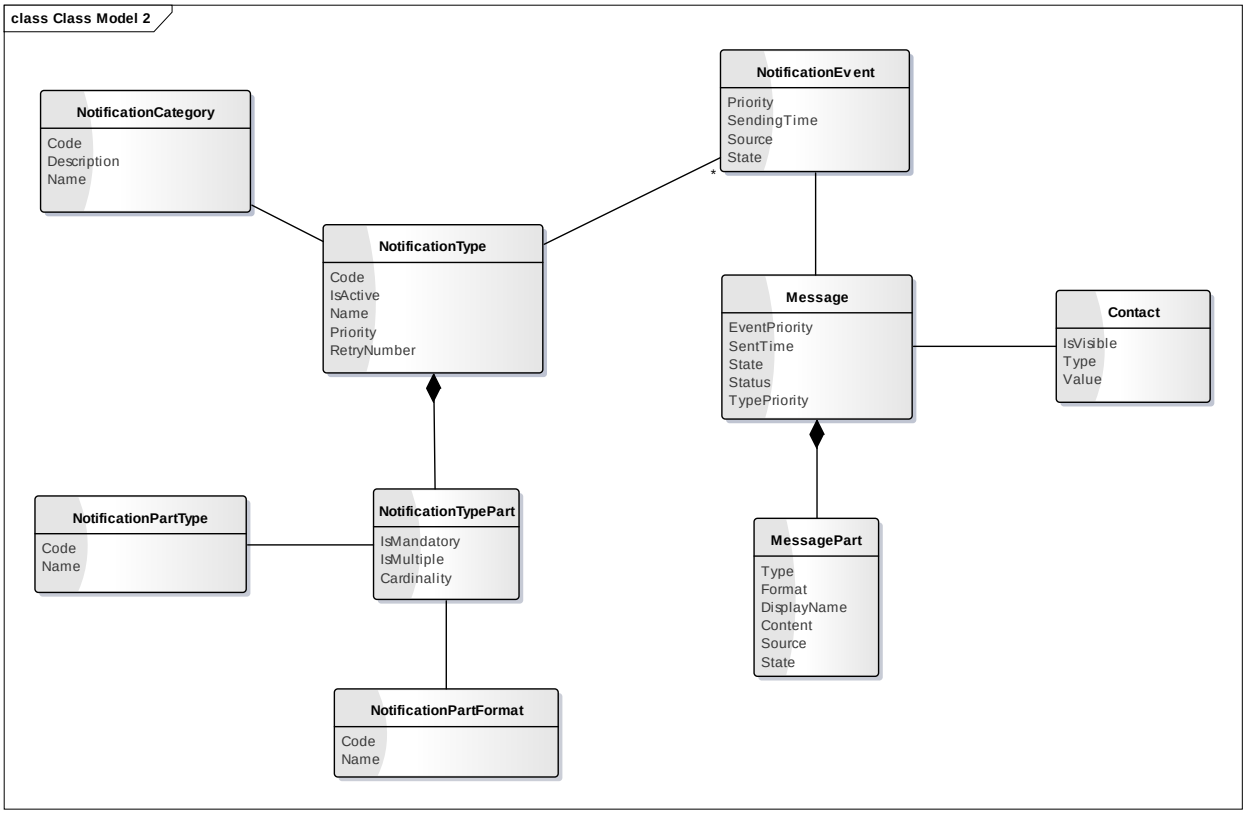


Figure 58 Notification

NotificationEvent is a concept that represents the input stream into the notification system and is a carrier of all necessary information (sender priority, data source, timestamp, etc.) for creating notification.

The states which the Notification Event object passing through are:

- **Initialized** – „INI“ (Events are formed and initialized)
- **Processing** – „PRO“ (Events are taken to the processing)
- **Managed** – „MNG“ (Events are processed and other objects(messages) in the system are created based on them).

The central concept of the Notification model is NotificationType that represents definition of specific notifications. NotificationType determines subsystem (FBOS Core, FBOS Service, external subsystems ...), input / output channel (SMS, Mail, Web ...), as well as the function that starts notification processing. Also, on the central concept is a predefined priority by type of notification.

NotificationType belongs to a particular category (NotificationCategory) and consists of several component parts (NotificationTypePart) which determine its structure. NotificationTypePart can be of a certain type (NotificationPartType) and format (NotificationPartFormat). Possible formats of the message parts are: PDF, HTML, TEXT, XML, and potential message parts are: Subject, Body, and Attachment. The model supports extensions, from the aspect of format as well as from the aspect of the message part type.

Message is a structure that represents a notification data and that will be sent to the proper addresses (Contact) and with a certain message, title, attachments (MessagePart), etc.

Message concept in its life cycle goes through several states:

- **Created** – “CRE” (message body is created, with priorities and contacts that will be sent to, as well as with its component parts without content)
- **Managed** – “MNG” (message and its component parts are entirely formed with full content, the message in this state is ready to be sent. Message enters this state when all its associated parts are placed in the status "MNG”)
- **Present** – “PRESNT” (message is queued)
- **Sent** – “SNT” (message is sent to the proper addresses)
- Also, parts of messages are going through adequate states:
- **Created** – “CRE” (message part body is created, with no suitable content)
- **Processing** – “PRO” (part of the message is taken for processing, where the transformation to its final content is made)
- **Managed** – “MNG” (part of the message has been processed, and has got its final look)

Based on sender priority and priority of the type that every message has, final priority for each message is determined.

Mapping Manager

Mapping model enables the concept transformation of different models, business or technology. Model concept is connected to the subsystem eGateway's meta base via attributes fwModelID and fwNamespace. These values uniquely determine a business or technological domain to which the concepts that are being mapped belong. The basic concept of the model is **Concept**, which can represent any concrete or abstract type (fwTypeID) described in the eGateway subsystem's metamodel. The definition of concept mapping is maintained through ConceptMapping, a recursive relationship (represented as a base class). This concept through IsCreator attribute defines whether the target concept is also being created through this mapping.

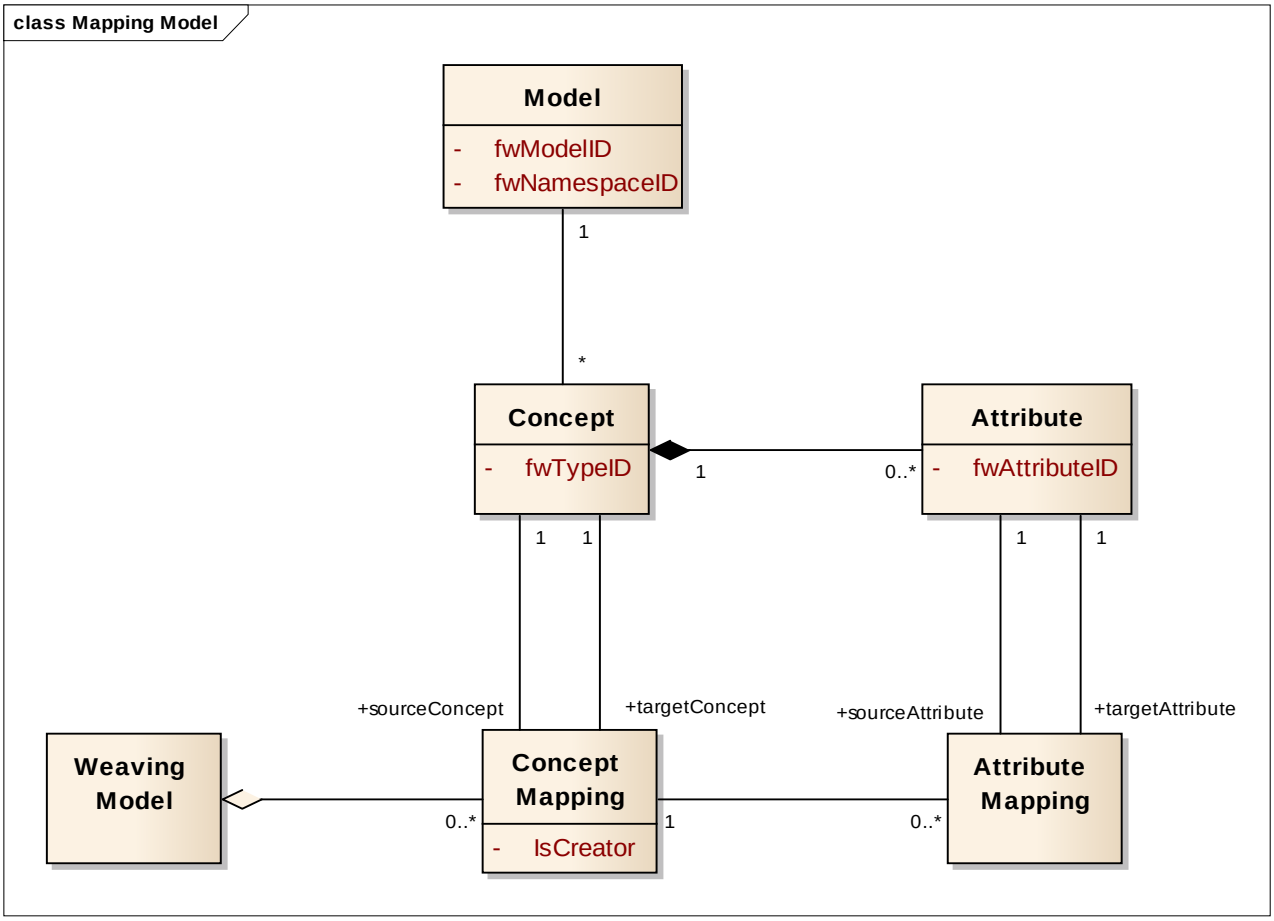


Figure 59 Mapping model

The attributes that are being mapped from the source to the target concept are defined through mapping in AttributeMapping object. The preferred mapping values are transferred based on the values associated with the **Attribute** structure with which Concept is connected. Each attribute is associated with the described objects' attributes using fwAttributeID attribute in the eGateway subsystem's metamodel. A set of concepts that are being mapped can be logically grouped through the WeavingModel concept.

Adapters Component

Adapter component aims to facilitate access and achieve communication to external systems and their data, regardless of whether the interface that external systems exhibit in the form of software components, Web services, file system, or stored procedures of some RDBMS.

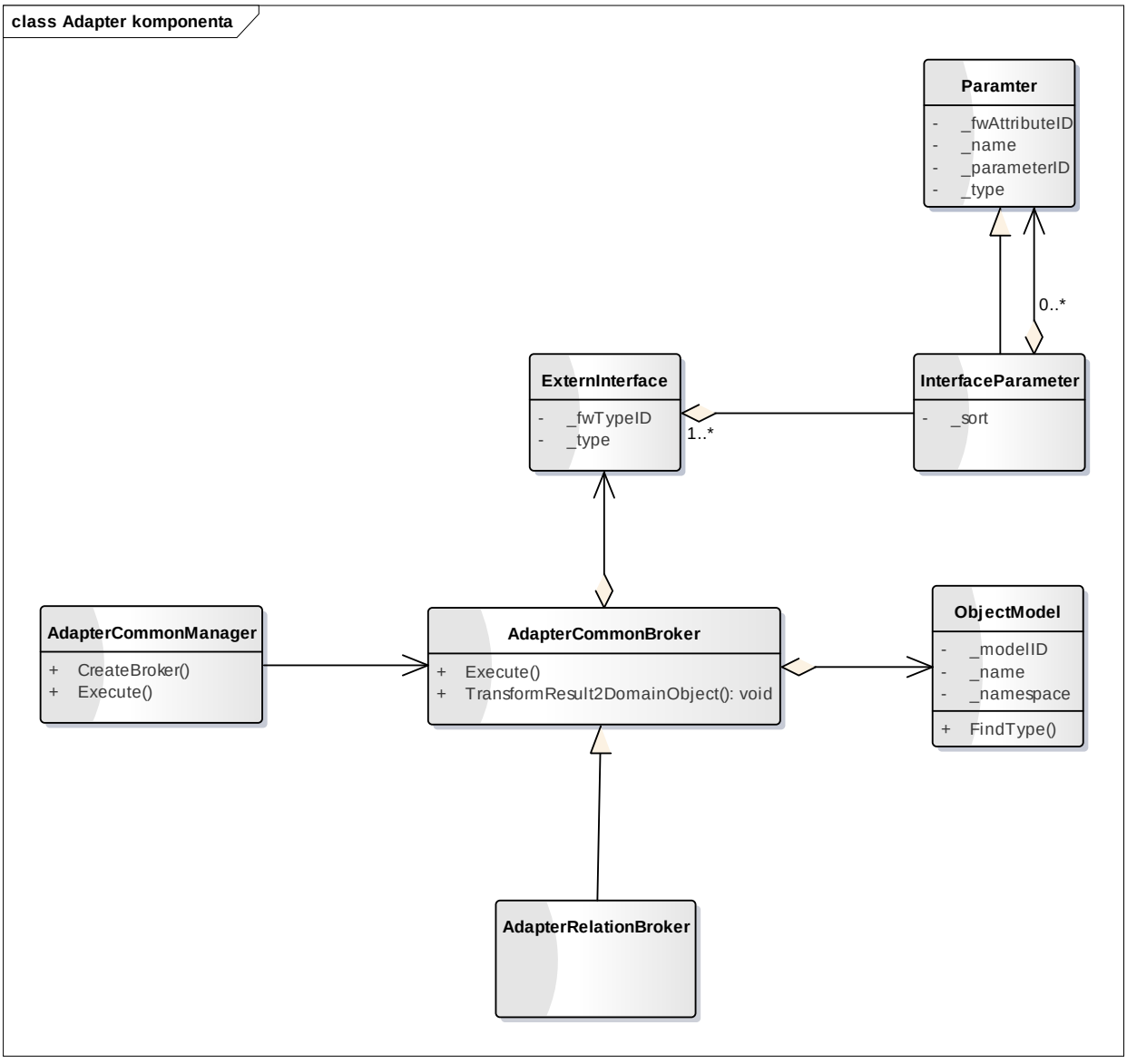


Figure 60 Object - oriented model of Adapter component

As depicted (Figure 39), shows the object - oriented model that is executed over the Adapter model which defines the external system interfaces. *ExternInterface* is a class that loads the interfaces together with the input and output parameters. *AdapterManager* has the responsibility to initiate execution of the interface and to create *AdapterBroker*, an abstract class that has its materialization through the corresponding brokers for each type of external system is shown *AdapterRelationBroker* to an external Core system. AdapterBroker has a link to the corresponding metamodel (ObjectModel) to connect our system domain objects specification with external systems interfaces. This means that it is possible to materialize domain objects based on data from an external data source. Supported external data sources are Oracle DB, external SQLServer, file system, XML documents, Web Services, etc.

Audit Log Component

Audit log is a component that ensures recording of system operation at the application level, such as the tracking of users who use the system, the functions that the user runs and objects he observes, creates or modifies. Also, the exact moment in which the user performs this action is recorded. The purpose of this component is to provide a review of activities that users performs while using the system.

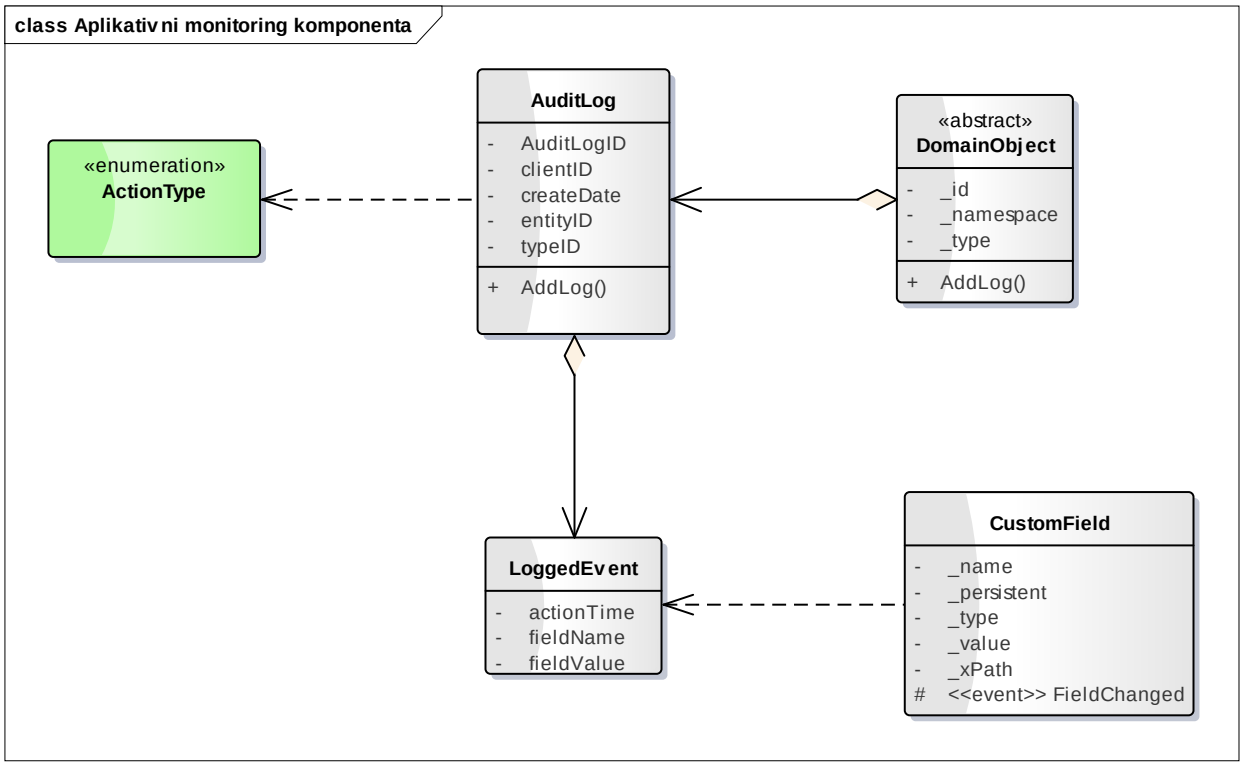


Figure 61 Aplication monitoring component

This component's functions are:

- recording of started system functions

- recording of objects and object attributes that are created / changed / viewed

- recording perform time

Exceptions Component

Exceptions is a component whose primary goal is to reliably determine the reason of reporting errors in the system, as well as the time and place of its occurrence. Also, the goal is appropriate processing of all errors, and showing the user a meaningful message that will focus on further possible actions.

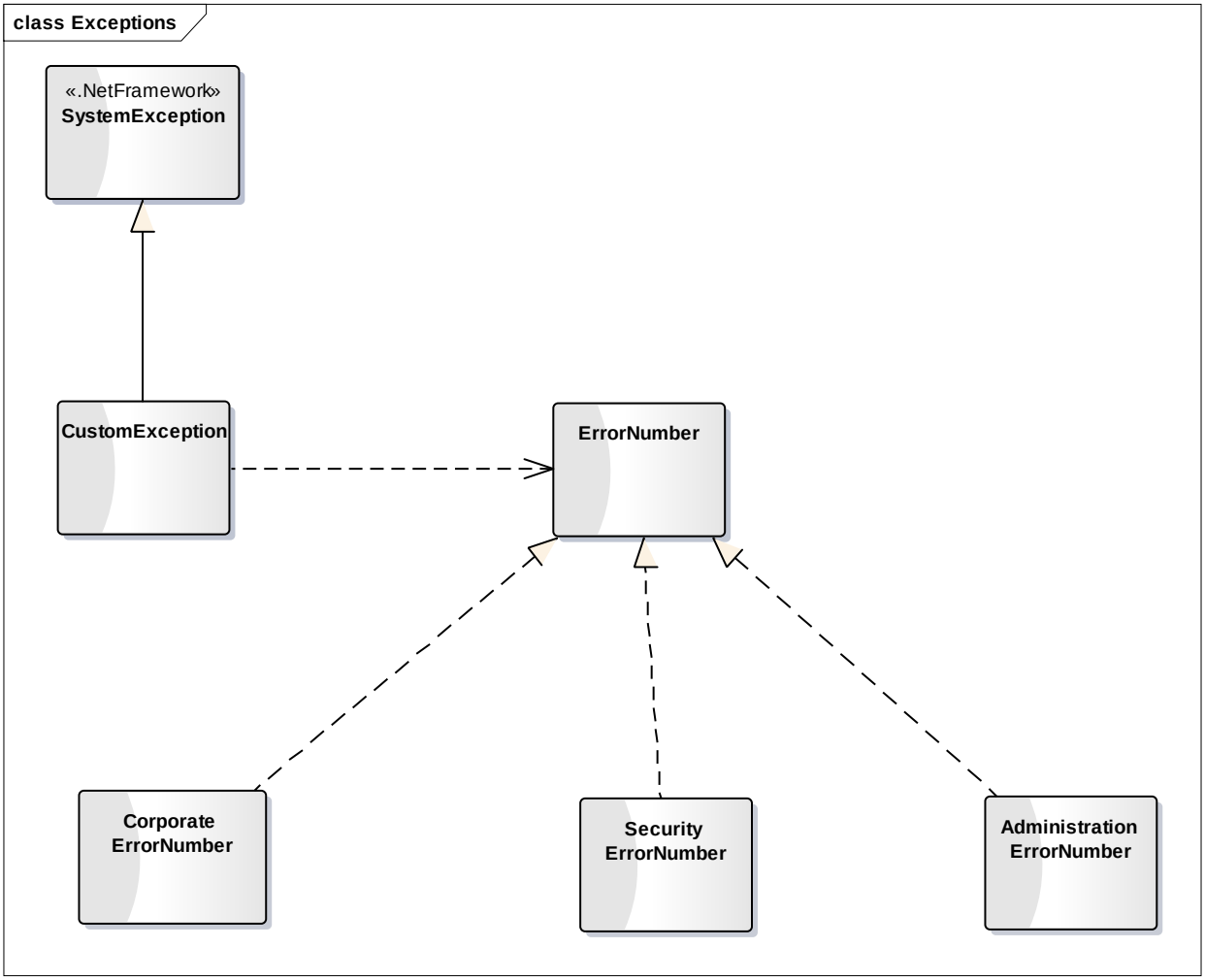


Figure 62 Object - oriented Exceptions model

As shown in the Figure 41, all exceptions in the system are of the CustomException type, in order to facilitate communication with external systems, as well as distributed communication within the system.

Data Types

Data Types is common component since it describes data types used in all applications and modules. There is already several defined data type models, such as C#, Java, Oracle, DB2, SQL Server. Data Types Component is capable to define new semantic data type models.

Non functional Requirements

Requirements or quality attributes during the system development

During the FBOS development, in order to provide Testability of the system, test/development environment (applications, databases etc.) will be provided thus ensuring possibility of testing of every functional and non-functional requirement;

Proposed software solution is adaptive, which provides a possibility of implementing new functionalities/properties into the system in an economically acceptable manner and with no adverse effects on existing functionalities;

Furthermore, proposed software solution enables Flexibility of the system and it's modifiability as the ability of the system to adjust to changes, e.g. regulations or business processes in an economically acceptable way;

As stated before, proposed software solution ensures Reusability, i.e. that some of the modules can be reused. This primarily refers to software modules.

As brifley described in Methodology section, FBOS development is secured by change management methodology that includes:

- Keeping records on all change management requests;
- Delivering requests for changes in an official form to official addresses of the parties;
- Change management requests containing all data necessary for decision – making;
- Decision – making related to acceptance of the requests at a proper project management level;
- Notification of all interested parties about the implemented change;
- Change implementation follow-up.

Usability

Proposed software solution enables primary target groups of users to use the system in a user – friendly manner. FBOS Usability will comprise the following attributes:

- simple learning curve to work with the software;
- efficiency and speed for trained users to achieve their goals using the software;

- forestalling all potential errors, a user may make using the software, and the ability of the system to recover efficiently in case of errors;
- User-friendly work with the software.

The usability of the system will be implemented through:

- Harmonization with business processes by following the users in their regular flow of activities in natural way;
- Intuitive user interface design by following best practice for the simplicity of web applications usage;

Most common transactions will be designed in such a way that they can be performed with the least number of interactions (mouse or keyboard clicks);

Rules and behavior of user interface will be consistent through the entire system, including windows, menus, and commands;

All system modules will have an intuitive user interface and consistent object concept (fields, drop – down lists, choice of options) so that users may use different pages/system screens in a simplified manner (e.g. display of available data, data-entry validity, error notifications);

The system will offer default values in all the fields for data entry, where reasonable. Default values may be fixed in advance, defined by the user or determined based on the context;

Proposed solution will have prepared valid data which will provide for the rapid entry of data, less sensitive to errors. The users will have the option of selecting appropriate elements from the list (the list shows appropriate code and description) or to insert the code directly. Once selected, an appropriate code and description is always visible to all data entries and query screens;

Double – check principle will be ensured for all the changes of reference data within a module for reference data management. Confirmation will be requested from the other (different) user. The module for managing reference data and all its clients use the original version of the record, until its modifications get verified, i.e. the modifications are pending until confirmed;

The system will prevent data redundancy, i.e. it should ensure that all the data entered are unique. Data redundancy prevention will be achieved by method of relation normalization (at least 3NF).

Reliability

FBOS will be reliable in order to perform its functions without interruptions during a specific time period within set conditions. We have proposed redundancy in order to assure High Availability of the system.

Performance

Response time of the system will not be longer than 5 seconds for all transactional use cases in the system.

Security

Proposed solution has several security levels thus ensuring data integrity protection and prevention from unauthorized data access or manipulation.

Proposed solution infrastructure will be accessible only through IPSec VPN.

Proposed solution will be internally isolated on all levels (virtualization, storage, network). It will be implemented usage of best practice concerning information access, in accordance to ISO27001.

By using a proper user's authentication and authorization system, defining rights and roles within a system, data encryption, implementation of security elements into the application, monitoring and recording all activities on the system itself, log management on data changes, etc., system will successfully protect belonging resources.

Data contained in the system will be protected in line with current legislation, particularly with Law on Personal Data Protection and GDPR recommendations.

The system will use HTTPS encrypted communication for exchange of data between clients and the central server.

The system will use HTTPS encrypted communication and username/password credentials for exchange of data between FBOS and the Business Registry Agency.

The system will support electronic identification of the user, whose level of security depends on the level of access to the system. Electronic identification protocols will be designed in a flexible manner, so that they can support different methods of checking user's authenticity, such as:

- User password, for the low-level security access,
- Qualified electronic certificates, for the high-level security access.

Proposed solution will use strong password policy and expiry of password functionality. System parameter will enable the System Administrator to configure the expiry of password policy. The proposed solution will ensure memorizing the old password, in order to prevent the user from using the same password multiple times. Furthermore, the proposed solution will ensure the

settings of session's expiry, and in case of user's inactivity for a certain time period, automatic sign out is activated.

Proposed solution may provide using of Qualified Certificates of all Certification Bodies established in Republic of Serbia.

Proposed solution will provide that certain users may only perform permitted operations and have the insight into certain data and system resources, in line with the roles assigned by the System Administrator.

Database Backup

The proposed solution will create security back-ups on a daily, weekly, and monthly level, outside the system's working hours, in a way that will not impact regular operation of the system.

For database backup proposed solution will use SQL Server included backup capabilities. It is a fine grain backup system which provides a lot of possibilities:

1. Backup compression
2. File and filegroup backup
3. Full, differential and transaction log backup
4. Backup to more files to reduce time for backup
5. Verification of the backup file

We will implement procedures for automatic backup, verification of backups and sending mail to operator if there is something wrong. We will fine tune backup process to be as fast as possible and to make possible fast restores when it is needed. We can also automatically generate restore scripts.

We will supply document with precise steps for fast recovery. Detailed maintenance plan will be provided by the end of the Inception phase.

Scalability

Proposed solution is fully scalable capable to cope with increased number of operation and users or greater number of services.

Proposed solution provides both, horizontal (Scale Out) and vertical (Scale Up) scalability.

Proposed solution has possibility of providing services to additional organizational units or organizations as well as possibility of providing additional services;

Interoperability

Proposed solution is fully interoperable with other existing systems in the public administration. Interoperability involves process interoperability, semantic interoperability and technical interoperability.

Proposed solution is based on Service Oriented Architecture, thus providing Openness of the system which ensures that other systems can connect with FBOS and modify the data or use its services. Interoperability and openness of the system will be implemented through the Integration Module.

Physical Architecture

Recommended infrastructure solution is fully compliant with requested HW specification:

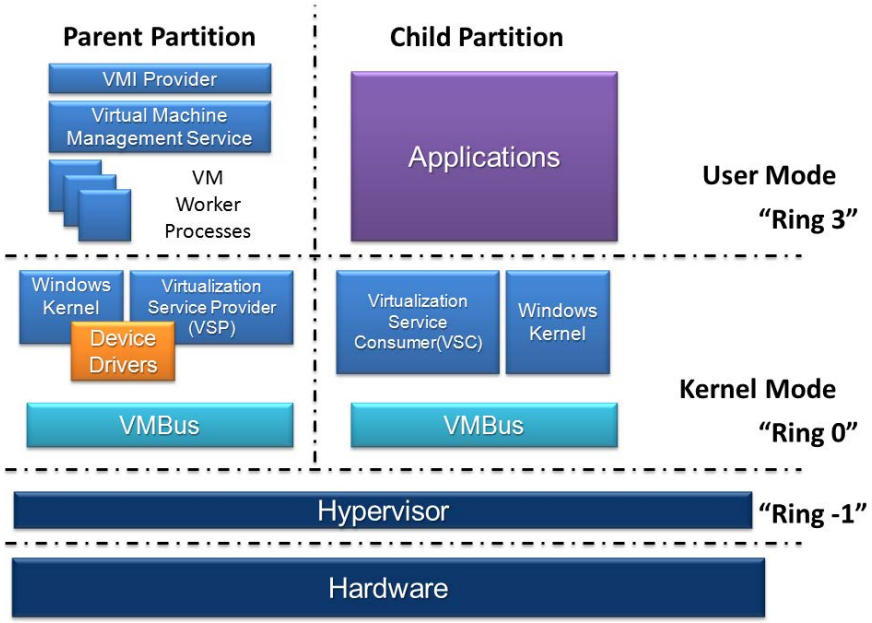
Hardware requirements

- The Subcontractor must accept to host the system during the full implementation phase as well as during the system warranty period. After that, (or earlier) the Project team will transfer the system to Project beneficiary.
- The subcontractor has to provide within its proposal the complete specification of server and client hardware platform, including needed virtualization software, and to harmonize the specification with the client representatives, without preferring any of manufacturers. The specified hardware should comply with the following requirements:
    - At each logical layer of the application there must be redundancy with a view of two servers (physical or virtual) allowing for load balance at the application level or failover at the level of databases management system;
    - Servers should be of rack or blade type with redundant components with multiple discs in RAID field and Storage Area Network system for functioning of failover cluster system at the level of databases management system;
    - For purposes of regular preparation of a spare data copy appropriate hardware system must be provided.
- The Subcontractor should within its proposal indicate the cost of a full list of required software and hardware resources needed to host the system.

Proposed physical architecture

Our SW solution is based on Microsoft technology and we propose architecture based on virtualized environment. Solution will consist of two environments, one for production and one for testing purposes.

All virtual machines for production will be highly available by utilizing clastering and/or load balancing. Test environments are not critical for normal business operations and will be single virtual machines.



Environement	Name of VM	vCPU	vRAM	PCS
Production	VM-DC	2	4	2
	VM-APP-PROD	8	16	2
	VM-DB-PROD	8	16	2
Test	VM-MW-TEST	4	8	1
	VM-DB-TEST	4	12	1

Virtualization

Natively with Microsoft Windows operating system comes Hyper-V Windows Server Virtualization and because of that we recommend it as virtualization platform.

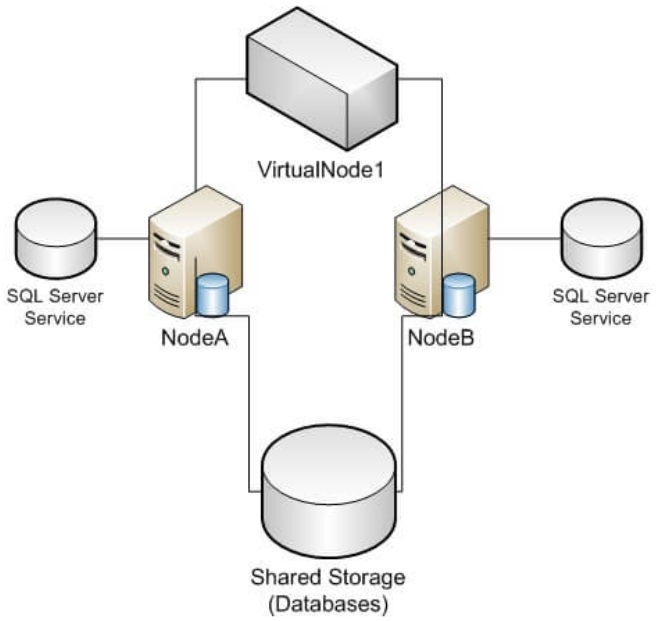
Isolation in Hyper-V is based on partitioning scheme. Generally speaking partition is logical unit of isolation supported by the hypervisor, in which operating systems execute. Main partition or root is a must and it is running Windows and all management and virtualization functions reside in it. All other partitions are child partitions or so called virtual machines. Each child partition has a virtual view of the HW and run in a virtual region that is private.

High availability for each virtual machine is achieved by utilizing Microsoft failover functionality.

Operating system – Quantity 8
Microsoft Windows Server 2019 Standard

Database subsystem will be implementid as higly available Microsoft SQL solution (each node is separate virtual machine) in active/standby configuration:

Database system – Quantity 8 cores
Microsoft Windows Server 2019 Standard



In order for our solution to work HW must have minimum following characteristics:

Server – Quantity 2
Minimum 2 processor system
Installed 2 processors each with: • Minimum 8 physical cores; • Minimum 3.0 GHz basic frequency; • Minimum 24MB L3 Cache; • Minimum DDR4 2666;
Minimum 128 GB installed memory DDR4 2666
Internal disk adapter (HBA - Host Bus Adapter) must have at least 2GB Battery Backed Write Cache (or equivalent technology) and must support RAID 1, 5, 6 protection level Minimum 3x 300 GB hard disks, organized in RAID 1, with minimum 1 stand-by hot-spare disk Internal disk adapter (HBA - Host Bus Adapter) of SAS type 6/12Gb/s for external storage connectivity
Ethernet: minimum 4 x 10Gbps RJ-45
Redundant hot-plug power supply Redundant hot-plug fans
Rack rails
Embedded full server management including virtual KVM console, LDAP integration, Virtual media
Ability to lock down configuration and firmware, protecting the server from inadvertent or malicious changes
UEFI secure boot with custom certificates
Included installation in existing 19" rack; Included installation, configuration and integration with the IT Infrastructure; Included all needed power and network cables.
Windows Operating System must be provided for all CPUs/Cores
Warranty: minimum 1 year

Storage – Quantity 1
Minimum 2 controller system
Minimum 6 GB cache memory per controller
2TiB usable capacity with one spare drive in RAID6 with SSD drives
Minimum 4xSAS 6/12Gb/s for host connectivity and 2x1Gb RJ 45 for management
No single point of failure design
Redundant hot-plug power supply
Thin provisioning
Web and CLI interface for management
Warranty: minimum 1 year

Network and security

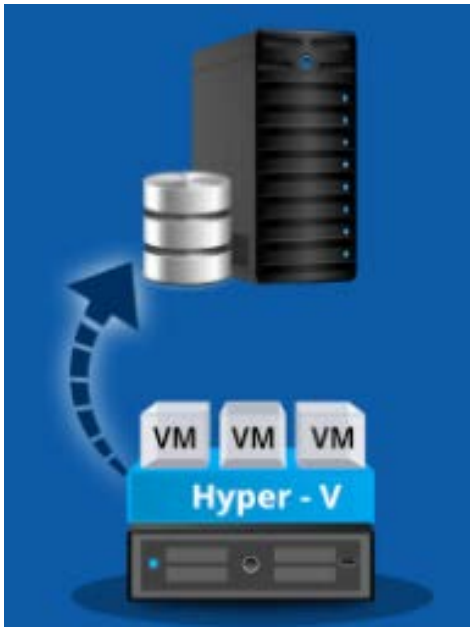
Network equipment – Quantity 2
Minimum 16 1/10Gb copper ports (RJ45)
Stackable or with VPC/MLAG/VLT support
Warranty: minimum 1 year

Since cybersecurity is one of the main concerns in the modern IT environment, the entire system must be protected with a firewall. Without it, the network is open to threats. A firewall keeps destructive and disruptive forces out and controls the incoming and outgoing network traffic based on security parameters that we can control and refine. In addition to supporting traditional firewall functions (packet filtering, network- and port-address translation (NAT), stateful inspection, virtual private networks (VPN)…), a firewall must support next-generation functions: application firewall, deep packet inspection (DPI), an intrusion prevention system (IPS)…

Firewall – Quantity 2
Minimum 8 1Gb copper ports (RJ45)
High Availability and/or Clustering
Firewall Services (stateful inspection, zone-based firewall)
Application Visibility
Network Address Translation (NAT)
VPN Features (Site-to-site, Remote Access)
Warranty: minimum 1 year

Backup support

Backup subsystem must support recommended infrastructure ecosystem and enable independent destination for backed data in order to be resilent and secure. It is necessary to have combination of backup software and data destination. For backup server we recommend separate virtual machine and for backup device separate NAS device with minimum capacity of 4TB usable space in redundant raid configuration.



Backup software will utilize Volume Shadow Copy Service (VSS) to provide transaction consistency for data being backed up. VSS will together with backup server coordinate the backup process.

Operating System – Quantity 1
Microsoft Windows Server 2019 Standard

NAS device – Quantity 1
Minimum two 1/10Gb copper ports (RJ45)
Acess over CIFS, NFS, iSCSI
Integration with Micrisoft Active Directory
Minimum 4TB usable space with NLSAS drives in redundant configuration
Warranty: minimum 1 year

Backup software – Quantity 1 for 4 sockets
Installation in virtual machine
Multi VM Instant recovery
Support for Deduplication
Aplication consistent backups
Transaction-level restore of Oracle databases and SQL Server databases
Support for full, incremental backups
Automatic recoverability testing of every backup and every replica
Integration with Hyper-V
Warranty: minimum 1 year