The OpenUp Process

Massimo Cossentino, Vincent Hilaire, and Valeria Seidita

Abstract

The Open Unified Process (OpenUp) is an iterative design process that structures the project lifecycle into four phases: Inception, Elaboration, Construction, and Transition. It is part of the Eclipse Process Framework and embraces a pragmatic, agile philosophy that focuses on the collaborative nature of software development. It is a tools-agnostic, low-ceremony process that can be extended to address a broad variety of project types. The project lifecycle provides stakeholders and team members with visibility and decision points throughout the project and makes them able to manage their work through micro-increments.

1 Introduction

OpenUp is a lean Unified Process that applies iterative and incremental approaches within a structured lifecycle. OpenUp embraces a pragmatic, agile philosophy that focuses on the collaborative nature of software development. It is a tools-agnostic,

M. Cossentino (✉)
Istituto di Reti e Calcolo ad Alte Prestazioni – Consiglio Nazionale delle Ricerche, Viale delle Scienze, 90128 Palermo, Italy
e-mail: cossentino@pa.icar.cnr.it

V. Hilaire
IRTES-SET, UTBM, UPR EA 7274, 90010 Belfort Cedex, France
e-mail: vincent.hilaire@utbm.fr

V. Seidita
Dipartimento di Ingegneria Chimica, Gestionale, Informatica, Meccanica, Viale delle Scienze, 90128 Palermo, Italy
e-mail: valeria.seidita@unipa.it

M. Cossentino et al. (eds.), *Handbook on Agent-Oriented Design Processes*,
DOI 10.1007/978-3-642-39975-6__15,

low-ceremony process that can be extended to address a broad variety of project types.

Personal efforts on an OpenUp project are organized in micro-increments. These represent short units of work that produce a steady, measurable pace of project progress (typically measured in hours or a few days).

The process applies intensive collaboration as the system is incrementally developed by a committed, self-organized team. These micro-increments provide an extremely short feedback loop that drives adaptive decisions within each iteration.

OpenUp divides the project into iterations: planned, time-boxed intervals typically measured in weeks. Iterations focus the team on delivering incremental value to stakeholders in a predictable manner. The iteration plan defines what should be delivered within the iteration. The result is a demo-able or shippable build. OpenUp teams self-organize around how to accomplish iteration objectives and commit to delivering the results. They do that by defining and "pulling" fine-grained tasks from a work items list. OpenUp applies an iteration lifecycle that structures how micro-increments are applied to deliver stable, cohesive builds of the system that incrementally progresses toward the iteration objectives.

OpenUp structures the project lifecycle into four phases: Inception, Elaboration, Construction, and Transition. The project lifecycle provides stakeholders and team members with visibility and decision points throughout the project. This enables effective oversight, and allows you to make "go or no-go" decisions at appropriate times. A project plan defines the lifecycle, and the end result is a released application.

It is worth to note that the OpenUp description largely uses the concept of work product slot. This is, indeed, one of the peculiarities of this process approach. The definition of work product slot may be found in [2]: "Work product slots are indirections for the inputs of tasks of a Practice that allow practices to be documented independent of any other practice, i.e. independent of the work products produced by other practices. Practice task refer to work product slots as inputs, rather than refer directly to specific work products." It is a little bit different from the concept of Work Product for which a specific SPEM 2.0 icon exists; we decided to use the same icon for the two concepts maintaining the name within brackets in the case of Work Product Slot as the OpenUp documentation does.

The description of the OpenUp process reported in this chapter is taken from the OpenUp website [1]. The documentation approach adopted in that website is quite different from the IEEE FIPA SC00097B standard adopted in this book but it has been possible to retrieve most of the necessary information. In order not introduce any personal interpretation of OpenUp, the authors of this chapter preferred not to report the information that may not be explicitly found in the website. This brought to the omission of a few details but it is coherent with the spirit of this book where the proposed chapters have been written by people who are the primary authors of the described process or they are at least deeply involved in it. In this case, this situation was not verified and therefore a specific care has been needed (Fig. 1).

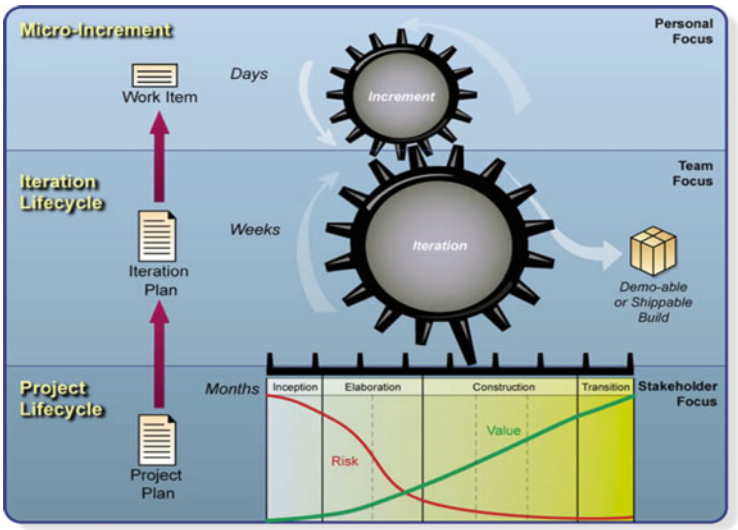

Fig. 1 An overview of the OpenUp process

1.1 The OpenUp Process Lifecycle

OpenUp is an iterative process with iterations distributed throughout four phases: Inception, Elaboration, Construction, and Transition.

Each phase may have as many iterations as needed (depending on the degree of novelty of the business domain, the technology being used, architectural complexity, and project size, to name a few factors).

To offer a quick start for teams to plan their iterations, OpenUp provides work breakdown structure (WBS) templates for each iteration, and a WBS template for an end-to-end process.

Iterations may have variable lengths, depending on project characteristics. One-month iterations are typically recommended because this timeframe provides:

- A reasonable amount of time for projects to deliver meaningful increments in functionality.
- Early and frequent customer feedback.
- Timely management of risks and issues during the course of the project.

In the following sections, all the aspects of OpenUp are described by using SPEM 2.0 [6] and the extensions proposed by Seidita et al. in [7]. Figure 2 shows the SPEM 2.0 icons the reader can find in the following figures (Fig. 3).

1.2 The OpenUp Process System Metamodel

The OpenUp website (see [1]) does not provide an explicit representation of the underlying system metamodel. For this reason, it has been preferred to limit

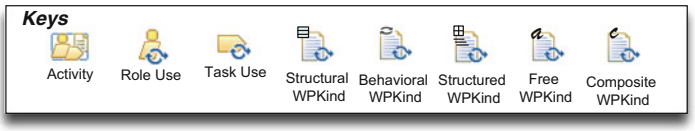

Fig. 2 The SPEM 2.0 Icons

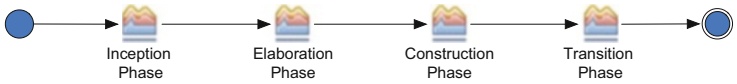

Fig. 3 The OpenUp phases

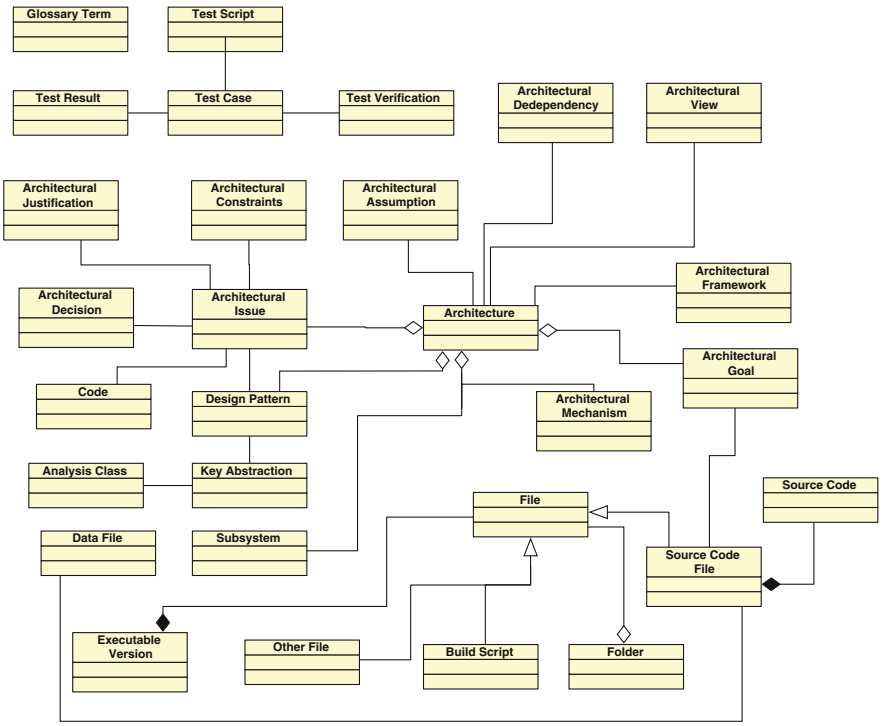

Fig. 4 The OpenUp System Metamodel—the first portion

the information reported in this section to what could be extracted from the website without any margin for misinterpretation. The elements of the metamodel have been deduced from the analysis of available work product descriptions. The following pictures (Figs. 4, 5, 6, and 7) only reports system metamodel elements whose relationships with others may be unambiguously deduced from work product descriptions; for readability reasons we split the whole figure in four ones.

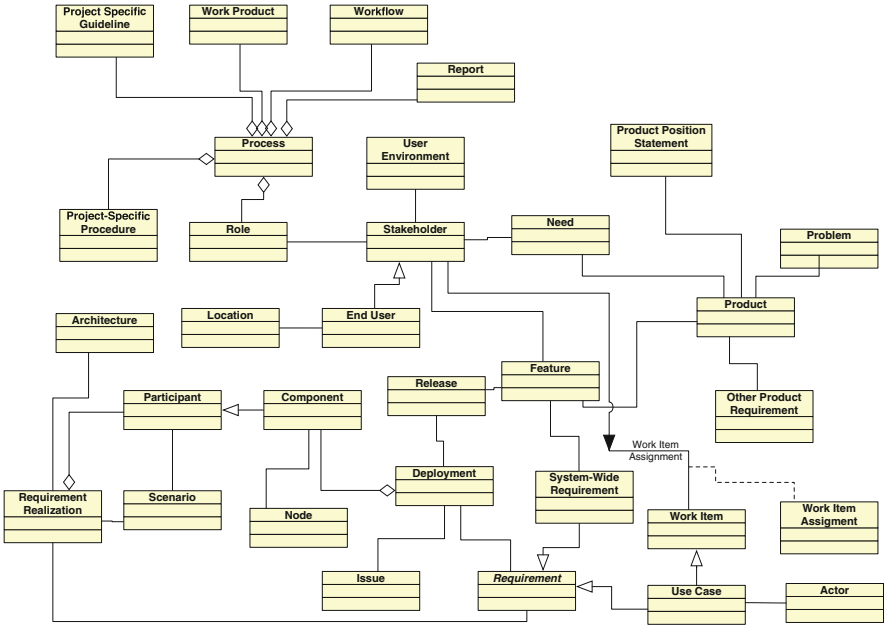

Fig. 5 The OpenUp System Metamodel—the second portion

1.2.1 Definition of the System Metamodel Elements

In the following table, the "Domain" column prescribed by the FIPA SC00097B specification has not been reported because there is no precise information on the OpenUp website about the allocation of concepts to domains like problem, solution and so on.

Table 1 only reports the elements for which a definition may be found in the OpenUp website. The other elements identified in the work product descriptions are reported below the table as a simple list.

List of system metamodel elements without definition: Analysis Class, Architecture, Architectural Issues, Architectural Mechanism, Build scripts, Communication Procedure, Component, Contingency Measure, Data files, Deployment, Design Pattern, End User, Implementation Element, Issue, Node, Other files, Product, Release, Requirement (abstract), Requirement Realization, Rollback, Source code files, Test Result, Workaround.

2 Phases of the OpenUp Process

2.1 The Inception Phase

The Inception phase is composed by the Inception iteration as described in Fig. 8. The process flow within the iteration is detailed in Fig. 9.

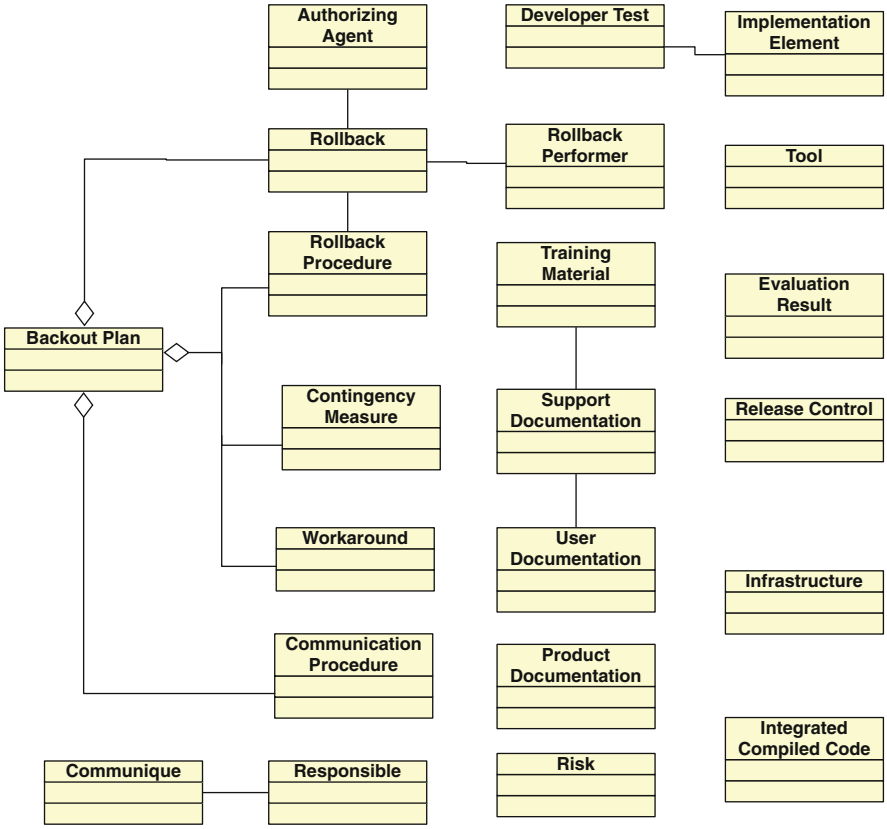

Fig. 6 The OpenUp System Metamodel—the third portion

The workflow inside each activity will be detailed in the following subsections (after the description of process roles). The Inception phase involves 8 different process roles, 24 work products as described in Figs. 10, 11, 12, and 13. The phase is composed of four activities as described in Fig. 10, each of them composed of one or more activities and tasks as described in Figs. 11, 12, 13, 14, and 15.

Details of activities shown in Fig. 10 are reported in Figs. 11, 12, 13, 14, and 15.

2.1.1 Process Roles

Eight roles are involved in the Inception phase, which are described in the following subsections.

Analyst

She/he is responsible for:

1. Detailing system-wide requirements
2. Detailing use-case scenario

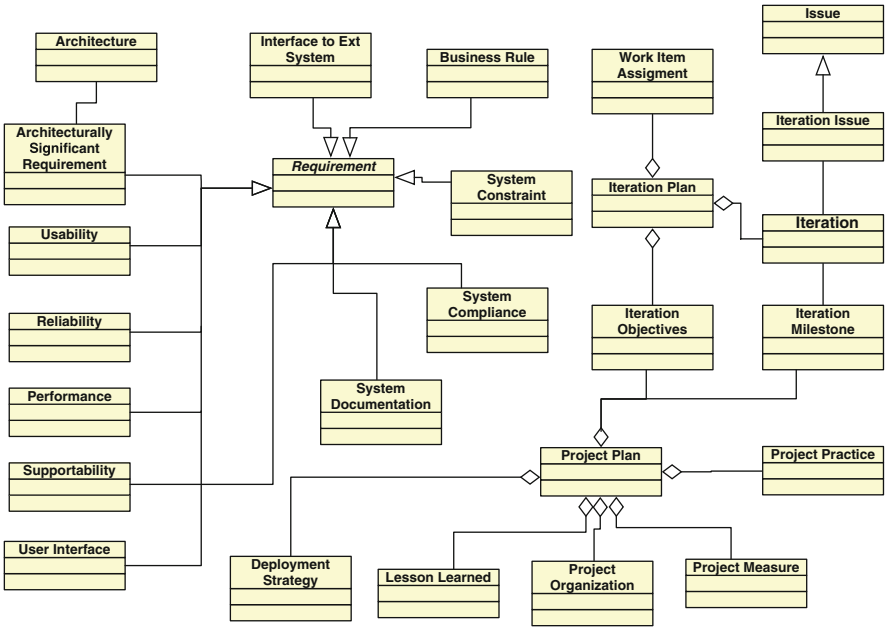

Fig. 7 The OpenUp System Metamodel—the fourth portion

3. Developing technical vision
4. Identifying and outlining requirements

She/he assists in:

1. Assessing results
2. Creating test cases
3. Envisioning the architecture
4. Managing iteration
5. Planning iteration
6. Planning project

Architect

She/he responsible for:

1. Envisioning the architecture.

She/he assists in:

1. Detailing system-wide requirements
2. Detailing use-case scenario
3. Developing technical vision
4. Identifying and outlining requirements
5. Managing iteration
6. Planning iteration
7. Plan project

Table 1 Definition of the system metamodel elements

Concept	Definition
Actor	To fully understand the system's purpose, you must know who the system is for, that is: Who will use the system? The answer to this question is: the Actors. An Actor is a role that a person or external system plays when interacting with the system. Instances of an Actor can be an individual or an external system; however, each Actor provides a unique and important perspective on the system that is shared by every instance of the Actor.
Communique	While there is no prescribed format for the release communications artifact, each communique should indicate the preferred delivery mechanisms (e.g., beeper notification, telephone calls, a posting to an internal release website, live or pre-recorded presentations by senior management, etc.) and generally answer the following questions: • Who are the parties (stakeholders) that are interested in knowing that a release to production has taken place? • What specifically (features, functions, components) has been placed into production? • Why is this release valuable to stakeholders and what business purpose does it serve? • Where is the product available (on which platforms, geographical locations, business units, etc.)? • How can the stakeholders access the system and under what circumstances? • When was the product released (or when will it be released if the release date is in the future)?
Design Element	The elements that will make up the implemented system. They contribute to define the abstractions of particular portions of the implementation. Design elements may be used to describe multiple static and dynamic views of the system for examination.
Developer Test	It covers all of the steps to validate a specific aspect of an implementation element. A developer test specifies test entries, execution conditions, and expected results. These details are identified to evaluate a particular aspect of a scenario
Envisioned Core Requirement	Define the quality ranges for performance, robustness, fault tolerance, usability, and similar characteristics that are not captured in the Feature Set.
Evaluation Results	Results of the iteration assessment that may be useful for improving the next one
Executable Version	The working version of the system or part of the system is the result of putting the implementation through a build process (typically an automated build script) that creates an executable version, or one that runs. This executable version will typically have a number of supporting files that are also considered part of this artifact.
Feature	The view of the stakeholders of the technical solution to be developed is specified in terms of her/his key needs and features. It also includes envisioned core requirements.
File	Files compose the executable version of the system.
Folder	Folders contain File
Glossary Term	Terms that are being used on the project so that everyone has a common understanding of them
Integrated Compiled Code	A release consists of integrated, compiled code that runs cleanly, independently, and in its entirety.
Term	Definition
Infrastructure	In reference to a release sprint, infrastructure refers to all the hardware, software, and network facilities necessary to support a deployed release. Infrastructure normally is defined as anything that supports the flow and processing of information in an organization. The infrastructure needed to support a release package normally includes: • Software, including: Operating systems and applications for servers and clients, Desktop applications, Middleware, Protocols • Hardware • Networks, including: Routers, Aggregators, Repeaters, Other transmission media devices that control movement of data and signals • Facilities

(continued)

Table 1 (continued)

Term	Definition
Iteration	An iteration is a set period of time within a project in which you produce a stable, executable version of the product, together with any other supporting documentation, install scripts, or similar, necessary to use this release. Also referred to as a cycle or a timebox.
Iteration Issue	An iteration is a set period of time within a project in which you produce a stable, executable version of the product, together with any other supporting documentation, install scripts, or similar, necessary to use this release. Also referred to as a cycle or a timebox.
Iteration Objectives	A few objectives should be written for the iteration, these will help guide the performers throughout the iteration. Also, assess at the end if those objectives have been achieved.
Iteration Plan	It helps the team to monitor the progress of the iteration, and keeps the results of the iteration assessment that may be useful for improving the next one. It include Milestones of the Iteration, task assignment and issues to be solved during the iteration.
Milestone	Milestones of an iteration show start and end dates, intermediate milestones, synchronization points with other teams, demos, and so on.
Need	Capabilities needed by stakeholder
Process	The process that a project is to follow in order to produce the project's desired results.
Product Documentation	It provides a detailed enough understanding of how the product operates and how it meets stated business goals and needs.
Project Plan	It describes how the project is organized, and identifies what practices will be followed. Additionally, it defines the parameters for tracking project progress, and specifies the high-level objectives of the iterations and their milestones.
Release Control	It identifies the requirements to which a release package must conform to be considered "deployable".
Responsible	Who will execute the communications when a successful release has been declared (normally the Deployment Engineer), as well as the timing and dependencies of the communiques.
Risk	A risk is whatever may stand in the way to success, and is currently unknown or uncertain. Usually, a risk is qualified by the probability of occurrence and the impact in the project, if it occurs.
Stakeholder	Stakeholders express their needs and requested features.
Support Documentation	Support documentation typically includes: • User manuals with work instructions, process descriptions, and procedures • Communications, training, and knowledge transfer deliverables • Support and operations manuals • Service information, including Help Desk scripts
System-Wide Requirement	System-wide requirements are requirements that define necessary system quality attributes such as performance, usability and reliability, as well as global functional requirements that are not captured in behavioral requirements artifacts such as use cases. System-wide requirements are categorized according to the FURPS+ model (Functional, Usability, Reliability, Performance, Supportability + constraints). Constraints include design, implementation, interfaces, physical constraints, and business rules. System-wide requirements and use cases, together, define the requirements of the system. These requirements support the features listed in the vision statement. Each requirement should support at least one feature, and each feature should be supported by at least one requirement.
Task Assignment	The task assignments for an iteration are a subset of all tasks on the Artifact: Work Items List
Test Case	A test case specifies the conditions that must be validated to enable an assessment of aspects of the system under test. A test case is more formal than a test idea; typically, a test case takes the form of a specification. It includes the specification of test inputs, conditions, and expected results for a system
Test Script	Test scripts implement a subset of required tests in an efficient and effective manner.

(continued)

Table 1 (continued)

Term	Definition
Test Verification	It reports that a set of tests was run
Tool	The tools needed for supporting the software development effort.
Training Material	Training materials that can be used to train end users and production support personnel might consist of: Presentation slides, Handouts, Job aids, Tutorials, On-line demos, Video vignettes, Lab exercises, Quizzes, Workshop materials, etc.
Use Case	Use cases are used for the following purposes: • To reach a common understanding of system behavior • To design elements that support the required behavior • To identify test cases • To plan and assess work • To write user documentation
User Documentation	User documentation might include all or parts of user manuals (electronic or paper-based), tutorials, frequently asked questions (FAQs), on-line Help Files, installation instructions, work instructions, operational procedures, etc.
Vision Constraint	Together with the Stakeholder Requests give an overview of the reasoning, background, and context for detailed requirements.
Work Item	Requests for additional capabilities or enhancement for that application. Work items can be very large in scope, especially when capturing requests for enhancements. To allow the application to be developed in micro-increments, work items are analysed and broken down into smaller work items so that they can be assigned to an iteration.

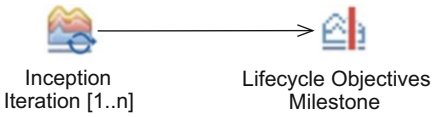

Fig. 8 The Inception iteration inside the Inception phase

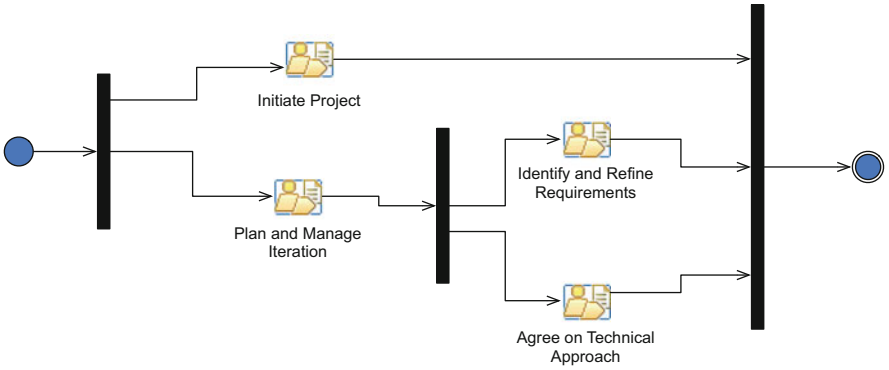

Fig. 9 The Inception phase flow of activities

Developer

She/he assists in

1. Assessing results
2. Creating test cases
3. Detailing use-case scenarios

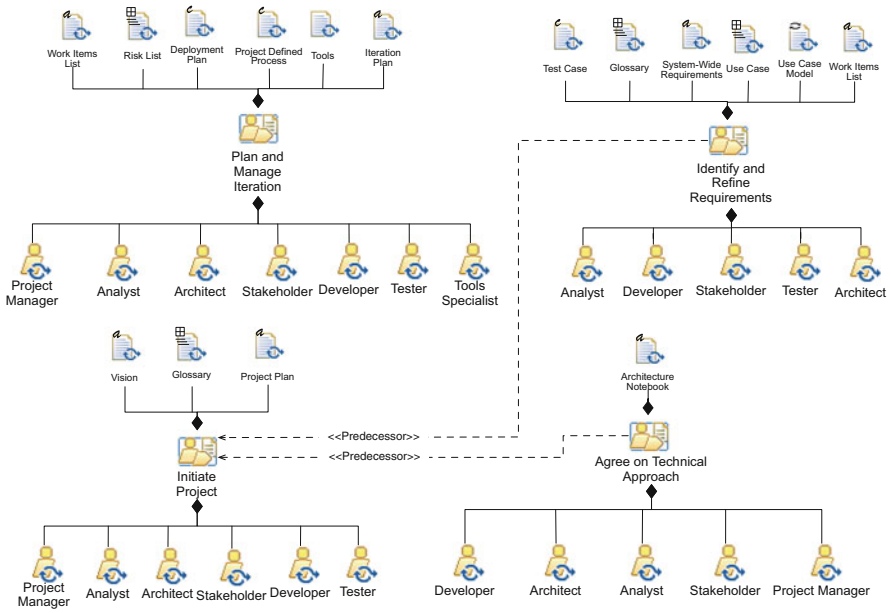

Fig. 10 The Inception phase described in terms of activities, output work products and involved stakeholders

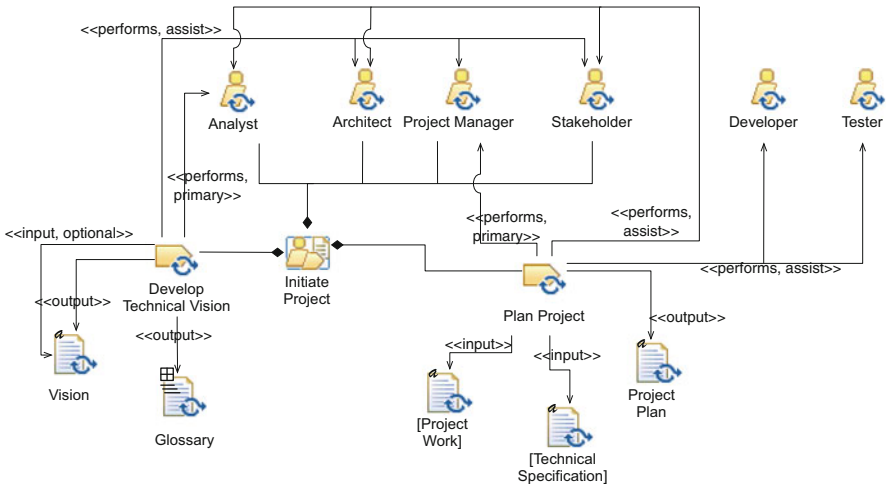

Fig. 11 The Initiate Project activity described in terms of tasks, roles and work products

4. Envisioning the architecture
5. Identifying and outlining requirements
6. Managing iteration
7. Outlining deployment plan

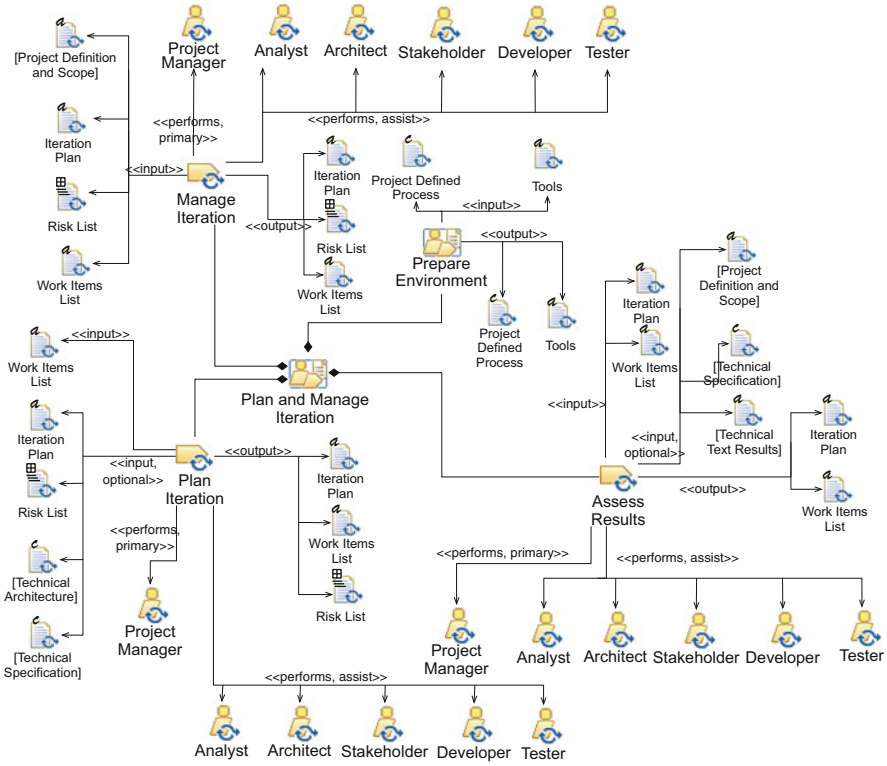

Fig. 12 The Plan and Manage Iteration activity described in terms of activities, tasks, roles and work products (activity Prepare Environment is detailed in Fig. 13)

8. Planning iteration
9. Planning project

Process Engineer

She/he is responsible for

1. Deploying the process
2. Tailoring the process

Project Manager

She/he is responsible for

1. Assessing results
2. Managing iteration
3. Planning iteration
4. Planning project

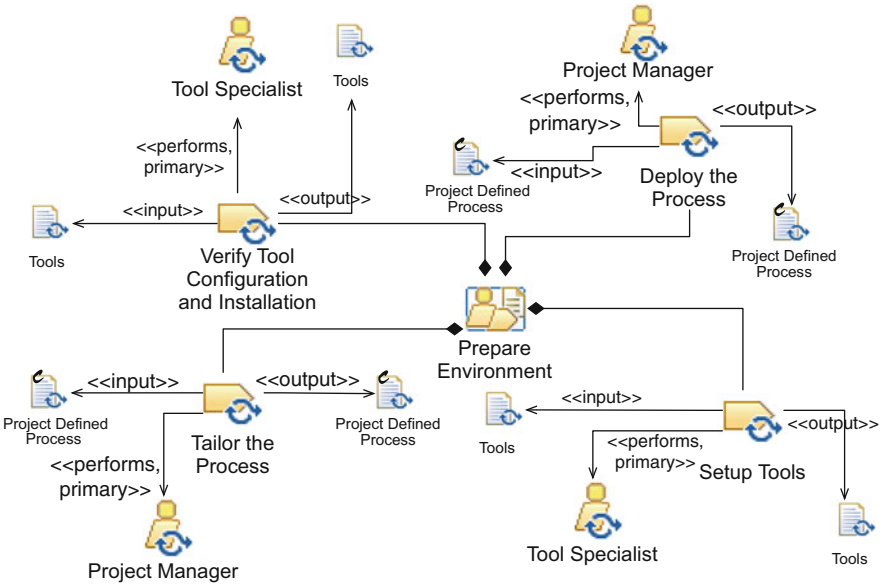

Fig. 13 The Prepare Environment activity of the Plan and Manage Iteration activity described in terms of tasks, roles, and work products

She/he assists in
1. Developing technical vision
2. Envisioning the architecture

Stakeholder

She/he assists in
1. Assessing results
2. Creating test cases
3. Detailing system-wide requirements
4. Detailing use-case scenarios
5. Developing technical vision
6. Envisioning the architecture
7. Identifying and outlining requirements
8. Managing iteration
9. Planning iteration
10. Planning project

Tester

She/he is responsible for
1. Creating test cases

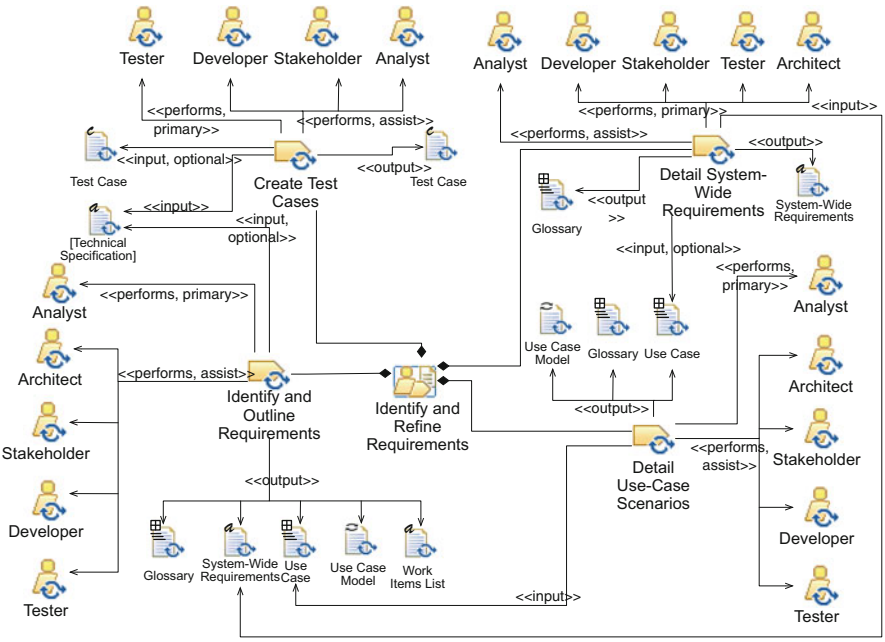

Fig. 14 The Identify and Refine Requirements activity described in terms of activities, tasks, roles, and work products

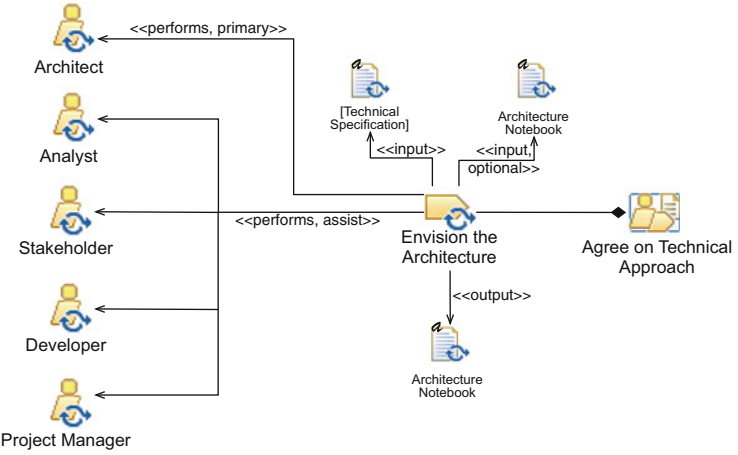

Fig. 15 The Agree on Technical Approach activity described in terms of activities, tasks, roles, and work products

She/he assists in

1. Assessing results
2. Detailing system-wide requirements
3. Detailing use-case scenarios

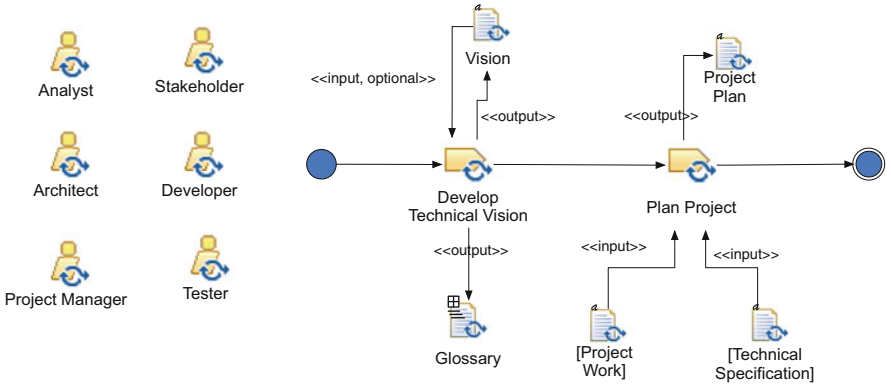

Fig. 16 The flow of tasks of the Initiate Project activity

Table 2 Initiate project—the task description

Activity	Task	Task description	Roles involved
Initiate Project	Initiate Project	The solution is proposed for a problem that everybody agrees on. Stakeholders collaborate with the development team to express and document their problems, needs, and potential features for the system to be, so the project team can better understand what has to be done.	Analyst (perform), Architect (assist), Project Manager (assist), Stakeholder (assist).
Initiate Project	Plan Project	Get stakeholder buy-in for starting the project and team commitment to move forward with it. This plan can be updated as the project progresses based on feedback and changes in the environment.	Project Manager (perform), Analyst (assist), Architect (assist), Developer (assist), Stakeholder (assist), Tester (assist).

4. Identifying and outlining requirements
5. Managing iteration
6. Planning iteration
7. Planning project

Tool Specialist
She/he is responsible for
1. Setting up tools
2. Verifying tool configuration and installation

2.1.2 Activity Details
The Inception phase includes four activities, which are described in the following subsections.

Initiate Project
The flow of tasks inside this activity is reported in Fig. 16, and the tasks are detailed in Table 2.

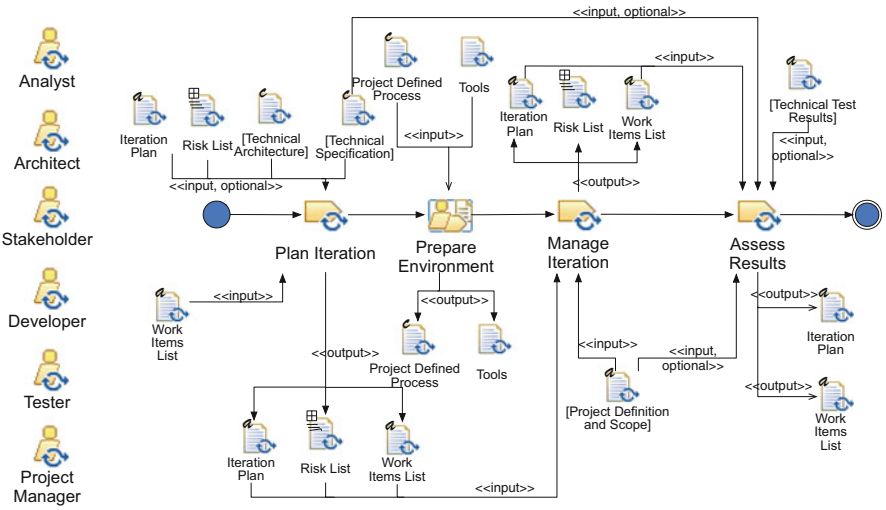

Fig. 17 The flow of tasks of the Plan and Manage Iteration activity

Table 3 Plan and manage iteration—the task description

Activity	Task	Task description	Roles involved
Plan and Manage Iteration	Plan Iteration	The purpose of this task is to identify the next increment of system capability, and create a fine-grained plan for achieving that capability within a single iteration.	Project Manager (perform), Analyst (assist), Architect (assist), Developer (assist), Stakeholder (assist), Tester (assist).
Plan and Manage Iteration	Manage Iteration	Help the team meet the iteration objectives and keep the project on track. Manage stakeholders' expectations as technical and practical discoveries are made during the project.	Project Manager (perform), Analyst (assist), Architect (assist), Developer (assist), Stakeholder (assist), Tester (assist).
Plan and Manage Iteration	Assess Results	Demonstrate the value of the solution increment that was built during the iteration and apply the lessons learned to modify the project or improve the process.	Project Manager (perform), Analyst (assist), Architect (assist), Developer (assist), Stakeholder (assist),Tester (assist).

Plan and Manage Iteration

The flow of tasks inside this activity is reported in Fig. 17, and the tasks are detailed in Table 3.

Prepare Environment

The flow of tasks inside this activity is reported in Fig. 18, and the tasks are detailed in Table 4.

Identify and Refine Requirements

The flow of tasks inside this activity is reported in Fig. 19, and the tasks are detailed in Table 5.

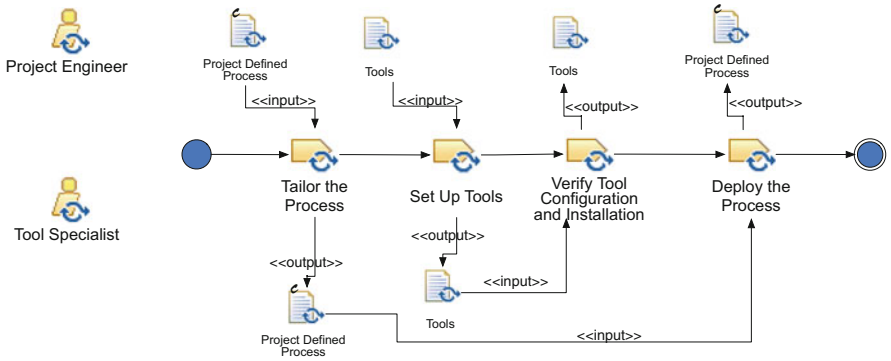

Fig. 18 The flow of tasks of the Prepare Environment activity

Table 4 Prepare environment—the task description

Activity	Task	Task description	Roles involved
Prepare Environment	Tailor the Process	The purpose of this task is to ensure that the project team has a defined process that meets their needs. The purpose of this task is to • Install the tools • Customize the tools • Make the tools available to the end users	Process Engineer (perform)
Prepare Environment	Set Up Tools	The purpose of this task is to • Install the tools • Customize the tools • Make the tools available to the end users	Tool Specialist (perform)
Prepare Environment	Verify Tool Configuration and Installation	The purpose of this task is to verify that the tools can be used to develop the system	Tool Specialist (perform)
Prepare Environment	Deploy the Process	The purpose of this task is to • Ensure that the project members are properly introduced to the process • Harvest any feedback on the process and refine the process, as necessary	Process Engineer (perform)

Agree on Technical Approach

The flow of tasks inside this activity is reported in Fig. 20, and the tasks are detailed in Table 6.

2.1.3 Work Products

The Inception phase generates fourteen work products. Their relationships with the system meta-model elements are described in Fig. 21.

This diagram represents the Iteration phase in terms of output Work Products. Each of these reports one or more elements from the OpenUp system metamodel; each system metamodel element is represented using an UML class icon (yellow filled) and, in the documents, such elements can be Defined, reFined, Quoted, Related or Relationship Quoted.

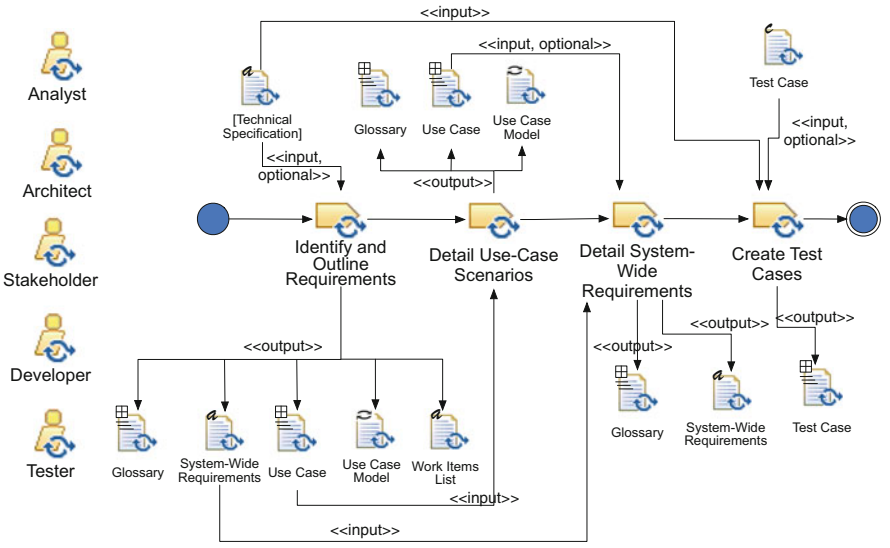

Fig. 19 The flow of tasks of the Identify and Refine Requirements activity

Table 5 Identify and refine requirements—the task description

Activity	Task	Task description	Roles involved
Identify and Refine Requirements	Identify and Outline Requirements	This task describes how to identify and outline the requirements for the system so that the scope of work can be determined.	Analyst (perform), Architect (assist), Developer (assist), Stakeholder (assist), Tester (assist).
Identify and Refine Requirements	Detail Use-Case Scenarios	This task describes how to detail use-case scenarios for the system.	Analyst (perform), Architect (assist), Developer (assist), Stakeholder (assist), Tester (assist).
Identify and Refine Requirements	Detail System-Wide Requirements	This task details one or more requirement(s) that do(es) not apply to a specific use case.	Analyst (perform), Architect (assist), Developer (assist), Stakeholder (assist), Tester (assist).
Identify and Refine Requirements	Create Test Cases	Develop the test cases and test data for the requirements to be tested.	Tester (perform), Analyst (assist), Developer (assist), Stakeholder (assist).

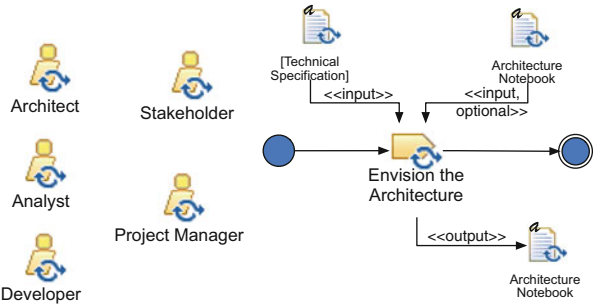

Fig. 20 The flow of tasks of the Agree on Technical Approach activity

Table 6 Agree on technical approach—the task description

Activity	Task	Task description	Roles involved
Agree on Technical Approach	Envision the Architecture	This task is where the 'vision" for the architecture is developed through analysis of the architecturally significant requirements and identification of architectural constraints, decisions and objectives.	Architect (perform), Analyst (assist), Developer (assist), Stakeholder (assist), Project Manager (assist).

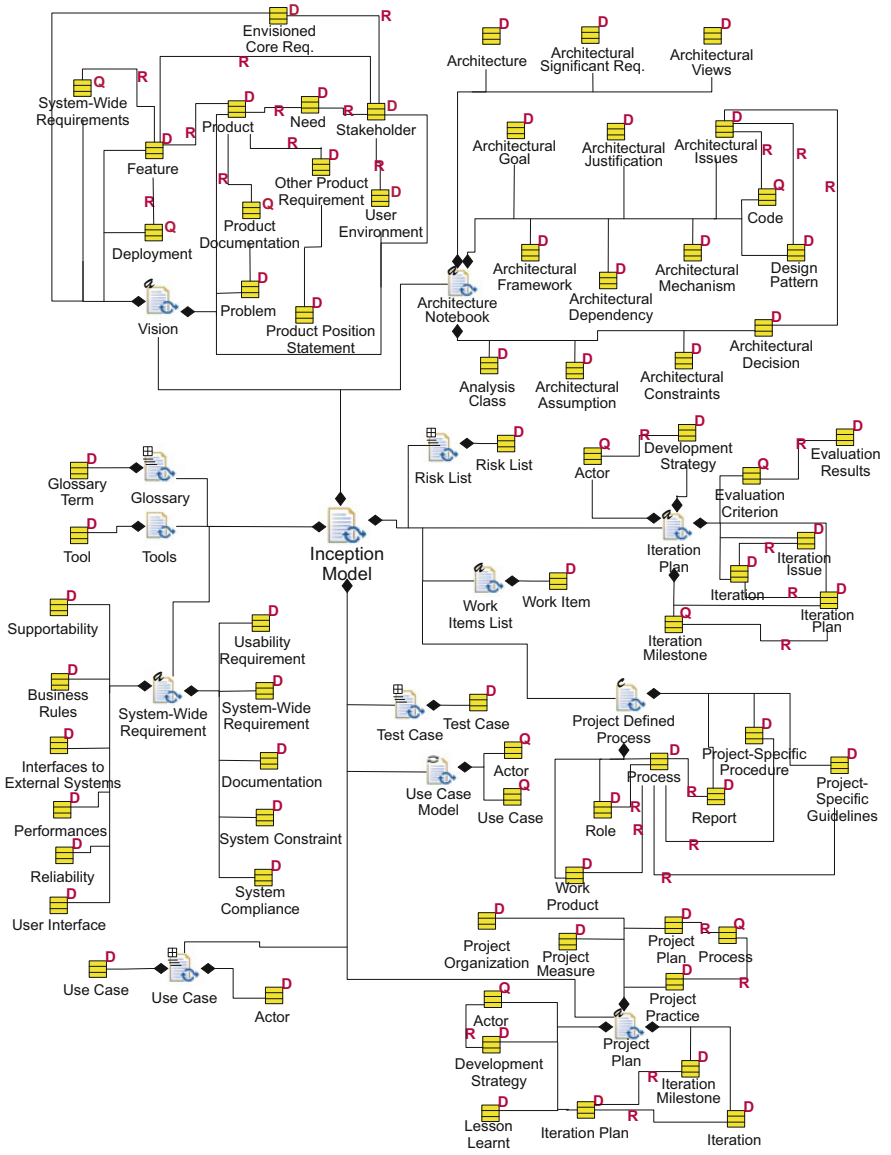

Fig. 21 The Inception phase documents structure

Work Product Kinds

Table 7 describes the work products of the Inception phase according to their kinds.

Work products are detailed in the following sections. No specific examples of notation use are reported since standard UML [3,4] is supposed to be adopted.

Architecture Notebook

The purpose of this artifact is to capture and make architectural decisions and to explain those decisions to developers. This artifact describes the Software Architecture. It provides a place for maintaining the list of architectural issues, along with the associated architectural decisions, designs, patterns, code documented (or pointed to), and so forth—all at appropriate levels to make it easy to understand what architectural decisions have been made and what remain to be made. It is helpful for architects to use this artifact to collaborate with other team members in developing the architecture and to help team members understanding the motivation behind architectural decisions so that those decisions can be robustly implemented. For example, the architect may put constraints on how data is packaged and communicated between different parts of the system. This may appear to be a burden, but the justification in the Architecture Notebook can explain that there is a significant performance bottleneck when communicating with a legacy system. The rest of the system must adapt to this bottleneck by following a specific data packaging scheme. This artifact should also inform the team members how the system is partitioned or organized so that the team can adapt to the needs of the system. It also gives a first glimpse of the system and its technical motivations to whoever must maintain and change the architecture later. At a minimum, this artifact should do these three things:

- List guidelines, decisions, and constraints to be followed
- Justify those guidelines, decisions, and constraints
- Describe the Architectural Mechanisms and where they should be applied

Team members who were not involved in those architectural decisions need to understand the reasoning behind the context of the architecture so that they can address the needs of the system. A small project should not spend a lot of time documenting the architecture, but all critical elements of the system must be communicated to current and future team members. This is all useful content:

- Goals and philosophy of the architecture
- Architectural assumptions and dependencies
- References to architecturally significant requirements
- References to architecturally significant design elements
- Critical system interfaces
- Packaging instructions for subsystems and components
- Layers and critical subsystems
- Key abstractions
- Key scenarios that describe critical behavior of the system

Table 7 Inception phase—work product kinds

Name	Description	Work product kind
Architecture Notebook	This artifact describes the rationale, assumptions, explanation, and implications of the decisions that were made in forming the architecture.	Free Text
Build	An operational version of a system or part of a system that demonstrates a subset of the capabilities to be provided in the final product.	Composite
Design	This artifact describes the realization of required system functionality and serves as an abstraction of the source code.	Composite
Developer Test	The Developer Test validates a specific aspect of an implementation element.	Structured
Glossary	This artifact defines important terms used by the project. The collection of terms clarifies the vocabulary used on the project.	Structured
Implementation	Software code files, data files, and supporting files (such as online help files) that represent the raw parts of a system that can be built.	Composite
Iteration Plan	A fine-grained plan describing the objectives, work assignments, and evaluation criteria for the iteration.	Free Text
Project Defined Process	This work product describes the process that a project is to follow in order to produce the project's desired results.	Composite
Project Definition and Scope	This slot serves as an abstraction of high-level artifacts that define the project and its scope. Typical examples of such artifacts could be a project definition, and a high-level project schedule identifying major milestones and major deliverables. Fulfilling Work Products: • Project Plan	Free Text
Project Plan	This artifact gathers all of the information required to manage the project on a strategic level. Its main part consists of a coarse-grained plan, identifying project iterations and their goals.	
Project Work	This slot serves as an abstraction for any type of work being done on the project. It could be represented as a work items list, an operational schedule, a work breakdown structure, and so on. Fulfilling Work Products: • Iteration Plan • Work Items List	Free Text
Risks List	This artifact is a sorted list of known and open risks to the project, sorted in order of importance and associated with specific mitigation or contingency actions.	Structured
System-Wide Requirements	This artifact captures the quality attributes and constraints that have system-wide scope. It also captures system-wide functional requirements.	Free Text
Technical Architecture	This slot serves as an abstraction of high-level artifacts that represent the documentation of the architecture. Fulfilling Work Products: • Architecture Notebook	Composite
Technical Specification	This slot serves as an abstraction of high-level artifacts that describe requirements, constraints, and goals for the solution. Fulfilling Work Products: • Glossary • System-Wide Requirements • Use Case • Use-Case Model • Vision	Composite
Technical Test Result	This slot serves as an abstraction of high-level artifacts that define the results of testing the hardware and software for the system being developed. Fulfilling Work Products: • Test Log	Free Text

(continued)

Table 7 (continued)

Name	Description	Work product kind
Test Case	This artifact is the specification of a set of test inputs, execution conditions, and expected results that you identify to evaluate a particular aspect of a scenario.	Structured
Test Log	This artifact collects the raw output that is captured during a unique run of one or more tests for a single test cycle run.	Free Text
Test Script	This artifact contains the step-by-step instructions that compose a test, enabling its run. Text scripts can take the form of either documented textual instructions that are manually followed, or computer-readable instructions that enable automated testing.	Structured
Tools	These work products are the tools needed to support the software development effort.	Free Text
Use Case	This artifact captures the system behavior to yield an observable result of value to those who interact with the system.	Structured Behavioral
Use-Case Model	This artifact presents an overview of the intended behavior of the system. It is the basis for agreement between stakeholders and the project team in regards to the intended functionality of the system. It also guides various tasks in the software development lifecycle.	Composite (Structured + Behavioral)
Vision	This artifact provides a high-level basis for more detailed technical requirements. It captures the technical solution by describing the high-level stakeholder requests and constraints that give an overview of the reasoning, background, and context for detailed requirements. The vision serves as input for communicating the fundamental "what and why" for the project and provides a strategy against which all future decisions can be validated. The vision should rally team members behind an idea and give them the context for decision-making in the requirements area. The vision must be visible to everyone on the team.	Free Text
Work Items List	This artifact contains a list of all of the scheduled work to be done within the project, as well as proposed work that may affect the product in this or future projects. Each work item may contain references to information relevant to carry out the work described within the work item.	Free Text

Build

The purpose of this work product is to deliver incremental value to the user and customer, and provide a testable artifact for verification. This working version of the system, or part of the system, is the result of putting the implementation through a build process (typically an automated build script) that creates an executable version. This version will typically have a number of supporting files that are also considered part of this artifact. This work product is almost always a product made up of numerous parts required to make the executable system. Therefore, a Build is more than just executable files; it also includes such things as configuration files, help files, and data repositories that will be put together, resulting in the product that the users will run.

Deployment Plan

The purpose of this work product is to capture, in one document, the unique information that will be consumed by deployment engineers before and during the deployment to production of a particular release package. The deployment

plan should contain the unique instructions for deploying a particular version of a product. By "unique instructions" we mean those things that are not part of a deployment engineer's normal procedures. Rather, they often are specific procedures and timing constraints that a deployment engineer should be aware of as they are rolling out a particular release. While a draft version of the deployment plan is typically developed by a development team, the deployment engineer is responsible for its contents and existence. A deployment plan normally consists of the following sections:

- The scope of the release and a general overview of the capabilities to be deployed
- The timing and dependencies for deploying components to various nodes
- The risks or issues associated with the release based on a risk assessment
- The customer organization, stakeholders, and end user community that will be impacted by the release
- The person or persons who have the authority to approve the release as "ready for production"
- The development team members responsible for delivering the release package to the Deployment Manager, along with contact information
- The approach for transitioning the release package to the Deployment Engineer, including appropriate communications protocols and escalation procedures
- The success criteria for this deployment; in other words, how will the Deployment Engineer know that the release is successful so it can report success

Design

The purpose of this work product is to describe the elements of the system so they can be examined and understood in ways not possible by reading the source code. This work product describes the elements that will make up the implemented system. It communicates abstractions of particular portions of the implementation. While architecture focuses on interfaces, patterns, and key decisions, the design fleshes out the technical details in readiness for implementation, or as part of implementation. This work product can describe multiple static and dynamic views of the system for examination. Although various views may focus on divergent, seemingly independent issues of how the system will be put together and work, they should fit together without contradiction. It is important that the author of this work product is able to analyse key decisions about the structure and behavior of the system and communicate them to other collaborators. It is also important that these decisions can be communicated at various levels of abstraction and granularity. Some aspects of the design can be represented by source code, possibly with some extra annotations. But more abstract representations of the design will be at a higher-level than source code. The more abstract representation could use various representation options. UML could be used either strictly or informally, it is the preferred notation based on its rich semantics and broad usage in the industry. Other techniques could be used to communicate the design. Or the design could use a mix of techniques as applicable.

This process does not govern whether to record these representations on a white board or to use a formal tool. But any representation, whether characterized as

formal or informal, should unambiguously communicate the technical decisions embodied by the design.

Developer Test

This artifact is used to evaluate whether an implementation element performs as specified. This artifact covers all of the steps to validate a specific aspect of an implementation element. For example, a test could ensure that the parameters of a method properly accept the uppermost and lowermost required values. A developer test specifies test entries, execution conditions, and expected results. These details are identified to evaluate a particular aspect of a scenario. When you collect developer tests for a specific implementation element, you can validate that the element performs as specified. The tests should be self-documenting so that it is clear upon completion of the test whether the implementation element has run correctly. Although there is no predefined template for this work product, and testing tools affect how the work product is handled, you should address the following issues:

- Setup
- Inputs
- Script
- Expected Results
- Evaluation Criteria
- Clean-Up

Suggestions and options for representing this work product: Suggestion: Automated code unit The most appropriate technique for running these tests is to use code that tests the implementation element scenarios and that you can run regularly as you update the system during development. When code is the sole form of the tests, ensure that the code is self-documenting. The code should document the specifications of the conditions you are testing and the setup or clean-up that is required for the test to run properly.

Option: Manual instructions: In some cases, you can use manual instructions. For example, when testing a user interface, a developer might follow a script, explaining the implementation element. In this case, it is still valuable to create a test harness that goes straight to the user interface. That way, the developer can follow the script without having to follow a complicated set of instructions to find a particular screen or page.

Option: Embedded code: You can use certain technologies (such as Java™ 5 Test Annotation) to embed tests in the implementation. In these cases, there will be a logical work product, but it will be assimilated into the code that you are testing. When you use this option, ensure that the code is self-documenting.

Glossary

These are the purposes of this artifact:

- To record the terms that are being used on the project so that everyone has a common understanding of them

- To achieve consistency by promoting the use of common terminology across the project
- To make explicit different stakeholders' use of the same terms to mean different things or different terms to mean the same thing
- To provide important terms to the Analysis and Design team

This artifact helps you avoid miscommunication by providing two essential resources:

- A central location to look for terms and abbreviations that are new to development team members
- Definitions of terms that are used in specific ways within the domain

Definitions for the glossary terms come from several sources, such as requirements documents, specifications, and discussions with stakeholders and domain experts.

Implementation

The purpose of this artifact is to represent the physical parts that compose the system to be built and to organize the parts in a way that is understandable and manageable. This artifact is the collection of one or more of these elements:

- Source code files
- Data files
- Build scripts
- Other files that are transformed into the executable system

Implementation files are represented as files in the local file system. File folders (directories) are represented as packages, which group the files into logical units.

Iteration Plan

The main objectives of the iteration plan are to provide the team with the following:

- One central place for information regarding iteration objectives
- A detailed plan with task assignments
- Evaluation results

This artifact also helps the team to monitor the progress of the iteration and keeps the results of the iteration assessment that may be useful for improving the next one. This artifact captures the key milestones of an iteration, showing start and end dates, intermediate milestones, synchronization points with other teams, demos, and so on. This artifact is also used to capture issues that need to be solved during the iteration. A few objectives should be written for the iteration, these will help guide the performers throughout the iteration. Also, assess at the end if those objectives have been achieved. The task assignments for an iteration are a subset of all tasks on the Artifact: Work Items List. Therefore, the iteration plan ideally references those work items. The evaluation criteria and iteration assessment information are captured in this artifact, so that it is possible to communicate results and actions from assessments. Work items assigned to an iteration do not necessarily have the same priority. When selecting work items from the Work Items List, the iteration plan may end up having work items with different priorities (for example, you assign the remaining high priority work items, plus a few mid-priority ones from the Work

Items List). Once work items have been assigned to the iteration, the team ensures that they can complete all work, regardless of original work item priorities. Deciding what to develop first on an iteration will vary across projects and iterations. The level of detail or formality of the plan must be adapted to what you need in order to meet these objectives successfully. The plan could, for example, be captured on the following places:

- A whiteboard listing the objectives, task assignments, and evaluation criteria
- A one-page document listing the objectives and evaluation criteria of the iteration, as well as referencing the Work Items List for assignments for that iteration
- A more complex document, supported by a Gantt or Pert chart in a project planning tool

Project Defined Process

The purpose of the project process is to provide guidance and support for the members of the project. "Information at your finger tips" is a metaphor that aligns well with the purpose of this work product. A project process typically describes or references the following items:

- What organizational processes and policies must be adhered to
- What standard process, if any, is being adopted by the project
- Any tailoring of the standard process, or deviations from policy mandates
- Rationale for tailoring and deviations
- Approvals for deviations
- Which work products are reviewed at which milestones, and their level of completion
- Guidelines and information that the project wants to use in addition to the information contained in the main process
- What reviews will be performed, and their level of formality
- What approvals are required, by whom, and when

Processes can be captured informally in documents, formally captured in a Method Composer configuration, or specified by configuring tools. Typically a project will use a combination of these: start with a Method Composer configuration, create a document to describe variations from this configuration and configure tools to support the process being followed.

Project Plan

The purpose of this artifact is to provide a central document where any project team member can find the information on how the project will be conducted. This artifact describes how the project is organized, and identifies what practices will be followed. Additionally, it defines the parameters for tracking project progress, and specifies the high-level objectives of the iterations and their milestones. The project plan allows stakeholders and other team members to understand the big picture and, roughly, when to expect a certain level of functionality be available. Update the plan as often as necessary, usually at the end of each iteration, in order to reflect changing priorities and needs, as well as record the lessons learned from the project. Create

and update the project plan in planning sessions that involve the whole team and appropriate project stakeholders in order to make sure that everybody agrees with it.

Risk List

The purpose of this work product is to capture the perceived risks to the success of the project. This list identifies, in decreasing order of priority, all the risks associated to a project. It serves as a focal point for project activities, and is the basis around which iterations are organized.

System-Wide Requirements

This artifact is used for the following purposes:

- To describe the quality attributes of the system, and the constraints that the design options must satisfy to deliver the business goals, objectives, or capabilities
- To capture functional requirements that are not expressed as use cases
- To negotiate between, and select from, competing design options
- To assess the sizing, cost, and viability of the proposed system
- To understand the service-level requirements for operational management of the solution

This artifact captures the quality attributes and constraints that have system-wide scope. It also captures system-wide functional requirements. This list should capture the critical and serious risks. If you find this list extending beyond 20 items, carefully consider whether they are really serious risks. Tracking more than 20 risks is an onerous task. A representation option for the risk list is to capture it as a section in the coarse-grained plan for the project. This means the coarse-grained plan has to be constantly revisited as you update risks. The fine-grained plans will contain only the tasks that you will be doing to mitigate risks in the short term.

Test Case

The purpose of this work product is

- To provide a way to capture test inputs, conditions, and expected results for a system
- To systematically identify aspects of the software to test
- To specify whether an expected result has been reached, based on the verification of a system requirement, set of requirements, or scenario

A test case specifies the conditions that must be validated to enable an assessment of aspects of the system under test. A test case is more formal than a test idea; typically, a test case takes the form of a specification. In less formal environments, you can create test cases by identifying a unique ID, name, associated test data, and expected results. Test cases can be derived from many sources, and typically include a subset of the requirements (such as use cases, performance characteristics and reliability concerns) and other types of quality attributes.

Test Log

The purpose of this work product is

- To provide verification that a set of tests was run

- To provide information that relates to the success of those tests

This artifact provides a detailed, typically time-based record that both verifies that a set of tests were run and provides information that relates to the success of those tests. The focus is typically on providing an accurate audit trail, which enables you to undertake a post-run diagnosis of failures. This raw data is subsequently analyzed to determine the results of an aspect of the test effort. Because this is a collection of raw data for subsequent analysis, it can be represented in a number of ways:

- For manual tests, log the actual results on a copy of the manual Test Script
- For automated tests, direct the output to log files that you can trace back to the automated Test Script
- Track raw results data in a test management tool

Test Script

Test scripts implement a subset of required tests in an efficient and effective manner.

Tools

These work products are the tools needed to support the software development effort.

Use Case

Use cases are used for the following purposes:

- To reach a common understanding of system behavior
- To design elements that support the required behavior
- To identify test cases
- To plan and assess work
- To write user documentation

A use case typically includes the following information:

- Name: The name of the use case.
- Brief Description: A brief description of the role and purpose of the use case.
- Flow of Events: A textual description of what the system does in regard to a use-case scenario (not how specific problems are solved by the system). Write the description so that the customer can understand it. The flows can include a basic flow, alternative flows, and subflows.
- Key scenarios: A textual description of the most important or frequently discussed scenarios.
- Special Requirements: A textual description that collects all of the requirements of the use case that are not considered in the use-case model, but that must be taken care of during design or implementation (e.g., non-functional requirements).
- Preconditions: A textual description that defines a constraint on the system when the use case starts.
- Post-conditions: A textual description that defines a constraint on the system when the use case ends.

- Extension points: A list of locations within the flow of events of the use case at which additional behavior can be inserted by using the extend-relationship.

You can document the use case as a use-case specification document or you can incorporate the use case in a use-case model. You can also use a requirements management tool to capture use cases and parts of use cases.

Use-Case Model

This artifact presents an overview of the intended behavior of the system. It is the basis for agreement between stakeholders and the project team in regards to the intended functionality of the system. It also guides various tasks in the software development lifecycle. Representation options include reports and diagrams from UML modeling tools, graphical representations created by using drawing tools, and drawings on whiteboards. Most of the information in the use-case model is captured in the use-case specifications.

Vision

This artifact provides a high-level basis for more detailed technical requirements. It captures the technical solution by describing the high-level stakeholder requests and constraints that give an overview of the reasoning, background, and context for detailed requirements. The vision serves as input for communicating the fundamental "what and why" for the project and provides a strategy against which all future decisions can be validated. The vision should rally team members behind an idea and give them the context for decision-making in the requirements area. The vision must be visible to everyone on the team. It is good practice to keep this artifact brief, so you can release it to stakeholders as soon as possible, and to make the artifact easy for stakeholders to read and understand. You can accomplish this by including only the most important features and avoiding details of requirements. Projects that focus on product development might extend the marketing section and include a more detailed product position statement that is based on their needs and research. Typically, the vision is represented in a document. If key stakeholder needs are captured in a requirements management tool, this part of the document can be generated by using reporting capabilities. If the vision serves a set of projects or an entire program, the overall vision might be divided into several vision work products. In this case, the vision of the program brings the visions together by providing program-specific content and referencing the subordinate visions.

Work Items List

The purpose of this artifact is to collect all requests for work that will potentially be taken on within the project, so that work can be prioritized, effort estimated, and progress tracked. This artifact provides a focal point for the entire team:

- It provides one list containing all requests for additional capabilities or enhancement for that application. Note that some of these requests may never be implemented, or be implemented in later projects.
- It provides one list of all the work to be prioritized, estimated, and assigned within the project. The risk list is prioritized separately.

- It provides one place to go to for the development team to understand what micro-increments need to be delivered, get references to material required to carry out the work, and report progress made.

These are the typical work items that go on this list:

- Use cases (and references to use-case specifications)
- System-wide requirements
- Changes and enhancement requests
- Defects
- Development tasks

Work items can be very large in scope, especially when capturing requests for enhancements, such as "Support Financial Planning" for a personal finance application. To allow the application to be developed in micro-increments, work items are analyzed and broken down into smaller work items so that they can be assigned to an iteration, such as a use-case scenario for "Calculate Net Worth". Further breakdown may be required to identify suitable tasks to be assigned to developers, such as "Develop UI for Calculate Net Worth". This means that work items often have parent/child relationships, where the lowest level is a specification and tracking device for micro-increments. This artifact should consist of the following information for each work item:

- Name and Description
- Priority
- Size Estimate
- State
- References

Assigned work items should also contain the following:

- Target Iteration or Completion Date
- Assignee
- Estimated Effort Remaining
- Hours Worked

Work Items should contain estimates. The recommended representation for the work items list is to capture it as a separate artifact, represented by a spreadsheet or database table. See Example: Work Items List. Alternatively, the work items list may be captured in tools such as project management, requirements management, or change request. In fact, the work items list may be spread over several tools, as you may choose to keep different types of work items in different repositories to take advantage of features in those tools. For example, you could use a requirements composition or management tool to track information about requirements, and use another tool to capture defects. Work items may start in one representation (such as in a spreadsheet) and move to more sophisticated tools over time, as the number of work items and the metrics you wish to gather grows more sophisticated. As part of the Iteration Plan, the plan typically references work items that are assigned to that iteration. If the team is capturing the iteration plan on a whiteboard, for example, the team may choose to reference high-level work items in the Work Items List that are assigned to the iteration, and maintain low-level child work items used to track day-to-day work only in an iteration plan.

Fig. 22 The elaboration iteration inside the elaboration phase

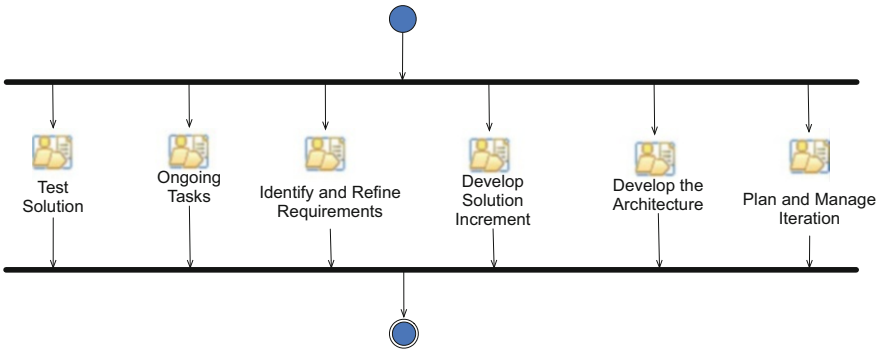

Fig. 23 The Elaboration phase flow of activities

2.2 The Elaboration Phase

The Elaboration starts when the first Milestone, Lifecycle Objectives Milestone, from Inception phase, is available. The Elaboration phase is composed by the Elaboration iteration as described in Fig. 22. The process flow within the iteration is detailed in Fig. 23.

The number and the length of each Elaboration iteration is dependent on, but not limited to, factors such as green-field development compared to maintenance cycle, unprecedented system compared to well-known technology and architecture, and so on. Typically, on the first iteration, it is better to design, implement, and test a small number of critical scenarios to identify what type of architecture and architectural mechanisms you need, so you can mitigate the most crucial risks. You also detail high-risk requirements that have to be addressed early in the project. You test enough to validate that the architectural risks are mitigated. During the subsequent iterations, you fix whatever was not right from the previous iteration. You design, implement, and test the remaining architecturally significant scenarios, ensuring that you check all major areas of the system (architectural coverage), so that potential risks are identified as early as possible [5].

The workflow inside each activity will be detailed in the following subsections (after the description of process roles). The Elaboration phase involves 10 different process roles, 29 work products and six activities (i.e., Plan and Manage Iteration, Identify and Refine Requirements, Develop the Architecture, Develop Solution Increment, Test Solution, Ongoing Tasks), as described in Fig. 24, each activity is

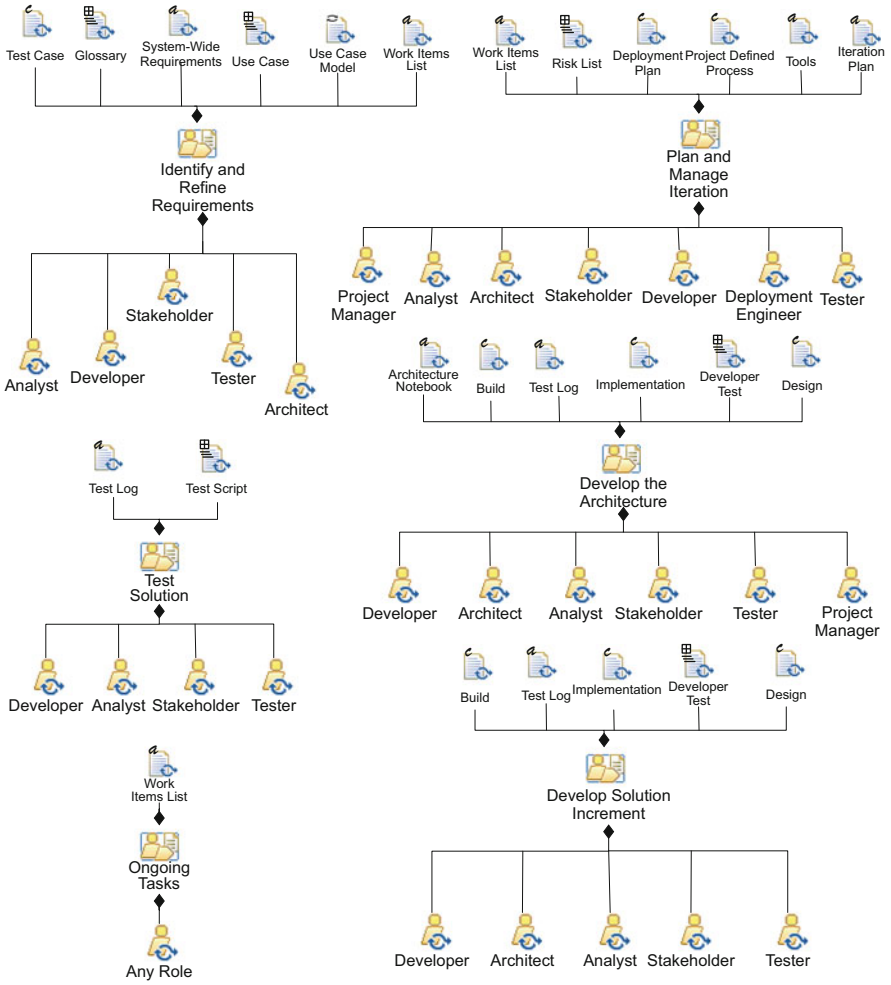

Fig. 24 The elaboration phase described in terms of activities, output work products and involved stakeholders

composed of one or more tasks/activities as described in Figs. 25, 26, 27, 28, and 29.

The description of the Prepare Environment activity in terms of tasks, roles, and work products is reported in Fig. 13. The description of the Identify and Refine Requirements activity in terms of tasks, roles, and work products is reported in Fig. 14.

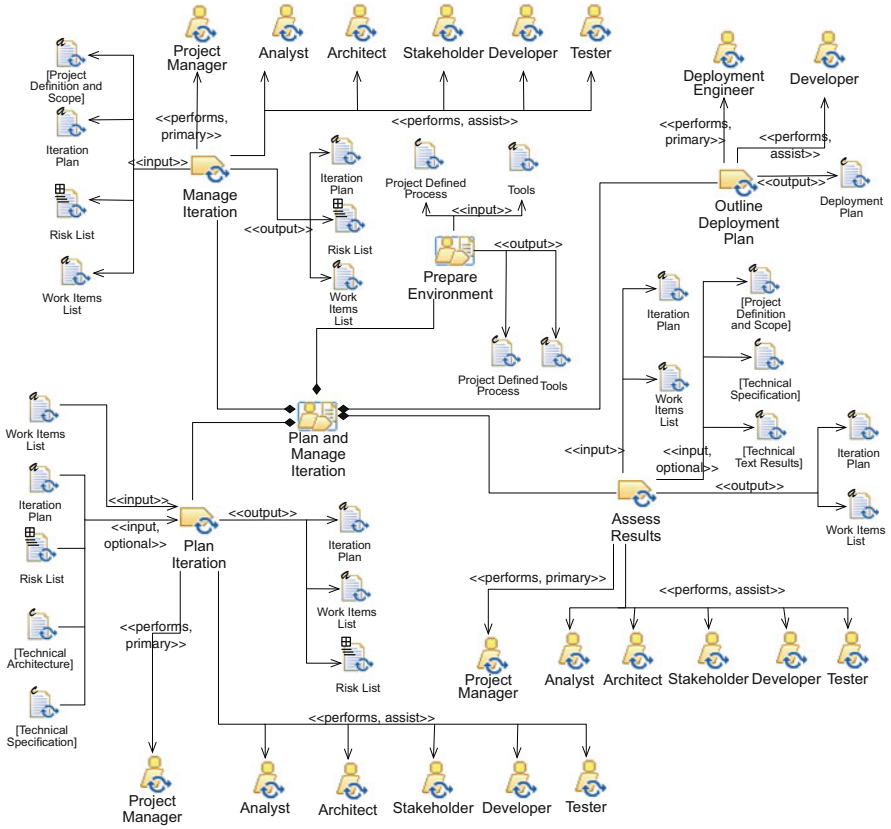

Fig. 25 The Plan and Manage Iteration activity described in terms of tasks, roles, and work products

2.2.1 Process Roles

Ten roles are involved in the Elaboration phase, which are described in the following subsections.

Analyst

She/he is responsible for the following tasks:

1. Detail System-Wide Requirements
2. Detail Use-Case Scenarios
3. Identify and Outline Requirements

She/he assists in the following tasks:

1. Assess Results
2. Create Test Cases
3. Design the Solution
4. Implement Tests

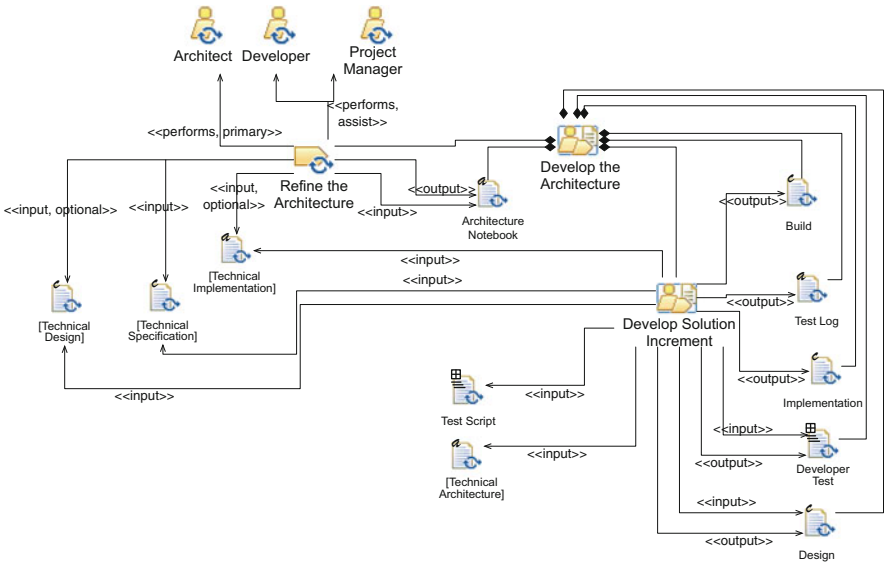

Fig. 26 The Develop the Architecture activity described in terms of tasks, roles, and work products

5. Manage Iteration
6. Plan Iteration

Any Role

She/he is responsible for the following tasks:

1. Request Change

Architect

She/he is responsible for the following task:

1. Refine the Architecture

She/he assists in the following tasks:

1. Assess Results
2. Design the Solution
3. Detail System-Wide Requirements
4. Detail Use-Case Scenarios
5. Identify and Outline Requirements
6. Manage Iteration
7. Plan Iteration

Deployment Engineer

She/he is responsible for the following task:

1. Outline deployment plan

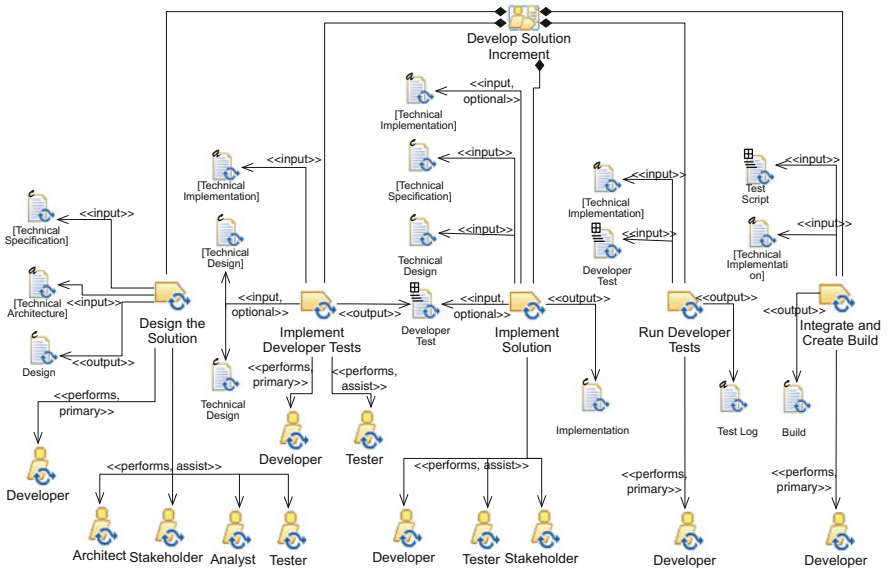

Fig. 27 The Develop Solution Increment activity described in terms of tasks, roles, and work products

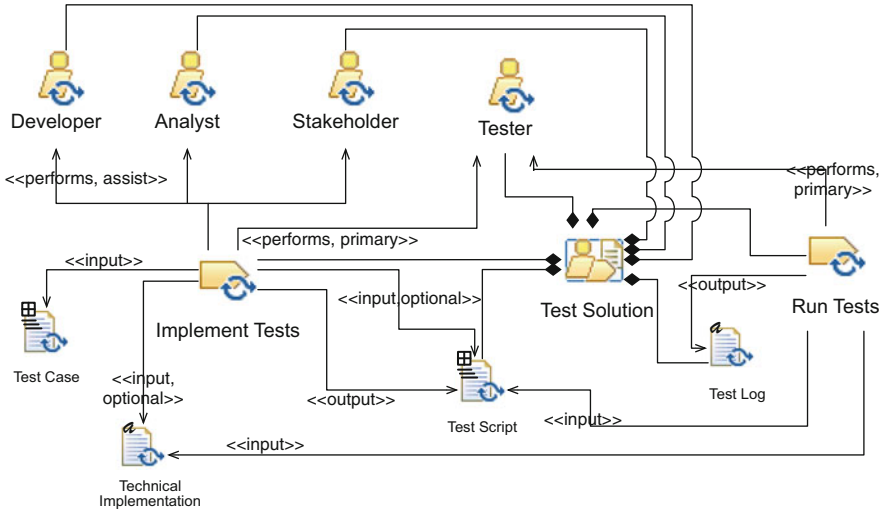

Fig. 28 The Test Solution activity described in terms of tasks, roles, and work products

Developer

She/he is responsible for the following tasks:

1. Design the Solution
2. Implement Developer Tests

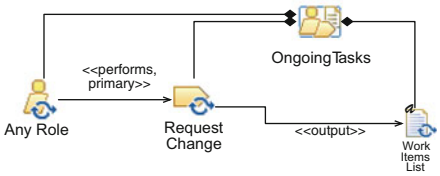

Fig. 29 The Ongoing Tasks activity described in terms of tasks, roles, and work products

3. Implement Solution
4. Integrate and Create Build
5. Run Developer Tests

She/he assists in the following tasks:

1. Assess Results
2. Create Test Cases
3. Detail System-Wide Requirements
4. Detail Use-Case Scenarios
5. Identify and Outline Requirements
6. Implement Tests
7. Manage Iteration
8. Outline Deployment Plan
9. Plan Iteration
10. Refine the Architecture

Process Engineer

She/he is responsible for the following tasks:

1. Deploy the process
2. Taylor the process

Project Manager

She/he is responsible for the following task:

1. Assess Results
2. Manage Iteration
3. Plan Iteration

She/he assists in the following task:

1. Refine the Architecture

Stakeholder

She/he assists in the following tasks:

1. Assess Results
2. Create Test Cases
3. Design the Solution
4. Detail System-Wide Requirements
5. Detail Use-Case Scenarios
6. Identify and Outline Requirements
7. Implement Solution

8. Implement Tests
9. Manage Iteration
10. Plan Iteration

Tester
She/he is responsible for the following tasks:
1. Create Test Cases
2. Implement Tests
3. Run Tests

She/he assists in the following tasks:
1. Assess Results
2. Design the Solution
3. Detail System-Wide Requirements
4. Detail Use-Case Scenarios
5. Identify and Outline Requirements
6. Implement Developer Tests
7. Implement Solution
8. Manage Iteration
9. Plan Iteration

Tool Specialist
She/he is responsible for the following tasks:
1. Set Up Tools
2. Verify Tool Configuration and Installation

2.2.2 Activity Details

The Elaboration phase includes six activities, which are described in the following subsections.

Plan and Manage Iteration
The goal of this activity is initiating the iteration, allowing team members to sign up for development tasks, monitoring and communicating project status to external stakeholders, and finally, identify and handling exceptions and problems. The flow of tasks inside this activity is reported in Fig. 30, and the tasks are detailed in Table 8.

Prepare Environment
See section "Prepare Environment" above.

Identify and Refine Requirements
See section "Identify and Refine Requirements" above.

Develop the Architecture
The goal of this activity is to develop the architecturally significant requirements prioritized for this iteration. The flow of tasks inside this activity is reported in Fig. 31, and the tasks are detailed in Table 9.

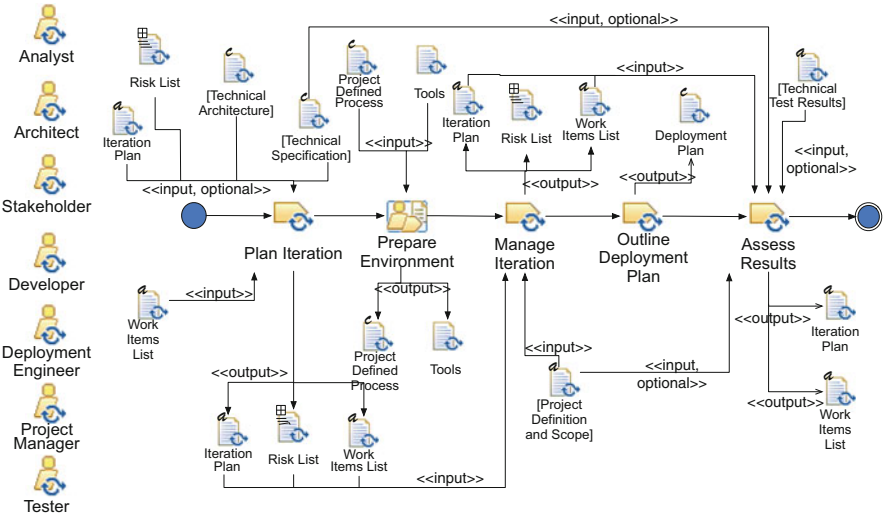

Fig. 30 The flow of tasks of the Plan and Manage Iteration activity

Table 8 Plan and manage iteration—the task description

Activity	Task	Task description	Roles involved
Plan and Manage Iteration	Plan Iteration	The purpose of this task is to identify the next increment of system capability, and create a fine-grained plan for achieving that capability within a single iteration.	Project Manager (perform), Analyst (assist), Architect (assist), Developer (assist), Stakeholder (assist), Tester (assist).
Plan and Manage Iteration	Manage Iteration	Help the team meet the iteration objectives and keep the project on track. Manage stakeholders' expectations as technical and practical discoveries are made during the project.	Project Manager (perform), Analyst (assist), Architect (assist), Developer (assist), Stakeholder (assist), Tester (assist). (perform), Domain Expert (assist)
Plan and Manage Iteration	Outline Deployment Plan	If a deployment plan for the project already exists, update it to reflect the nature of this release. If this document does not exist, develop a deployment plan to indicate how this release will be rolled out to the production environment.	Deployment Engineer (perform), Developer (assist).
Plan and Manage Iteration	Assess Results	Demonstrate the value of the solution increment that was built during the iteration and apply the lessons learned to modify the project or improve the process.	Project Manager (perform), Analyst (assist), Architect (assist), Developer (assist), Stakeholder (assist), Tester (assist).

Develop Solution Increment

The goal of this activity is to design, implement, test, and integrate the solution for a requirement within a given context. The flow of tasks inside this activity is reported in Fig. 32, and the tasks are detailed in Table 10.

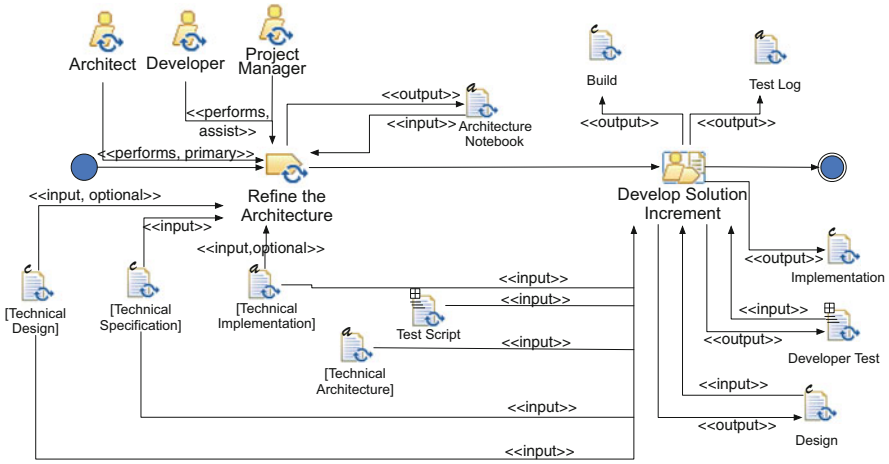

Fig. 31 The flow of tasks of the Develop the Architecture activity

Table 9 Develop the architecture—the task description

Activity	Task	Task description	Roles involved
Develop the Architecture	Develop Solution Increment	• For developers: To create a solution for the work item for which they are responsible • For project managers: To have a goal-based way of tracking project status	Developer (Primary), architect (additional), Analyst (additional), Stakeholder (additional), Tester (additional)
Develop the Architecture	Refine the Architecture	To make and document the architectural decisions necessary to support development.	Architect (Primary), Developer (additional), Project Manager (additional)

Test Solution

The goal of this activity is to test and evaluate the developed requirements from a system perspective. The flow of tasks inside this activity is reported in Fig. 33, and the tasks are detailed in Table 11.

Ongoing Tasks

The goal of this activity is to perform ongoing tasks that are not necessarily part of the project schedule. The flow of tasks inside this activity is reported in Fig. 34, and the tasks are detailed in Table 12.

2.2.3 Work Products

The Elaboration phase generates thirteen work products. Their relationships with the system metamodel elements are described in Fig. 35.

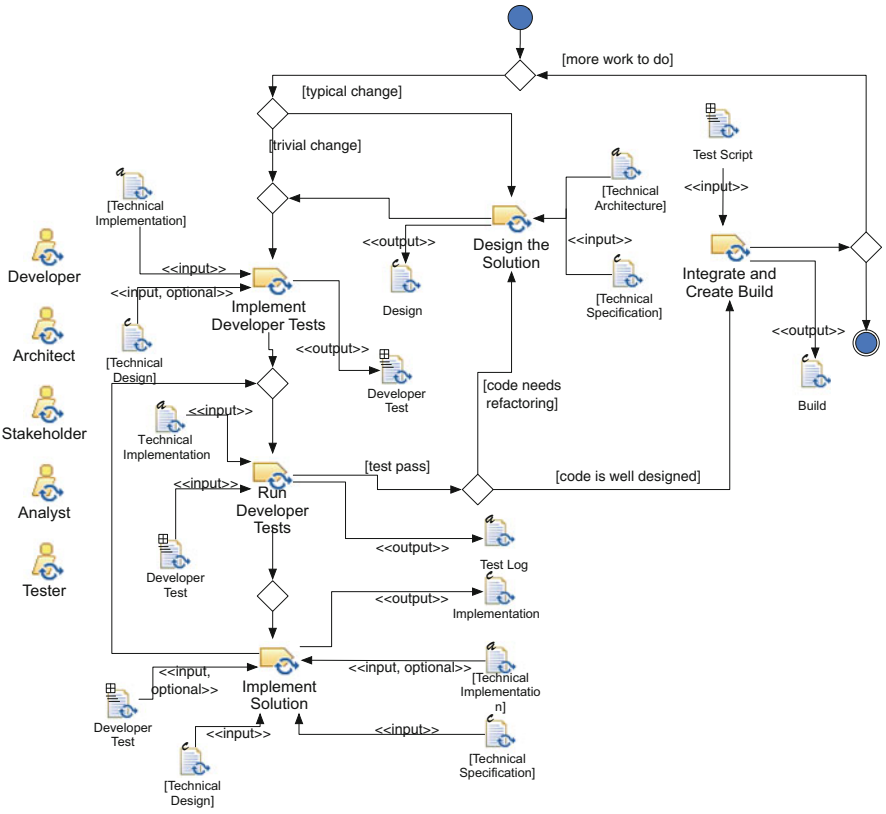

Fig. 32 The flow of tasks of the Develop Solution Increment activity

Work Product Kinds

Table 13 describes the work products of the Elaboration phase according to their kinds.

Architecture Notebook

See section "Architecture Notebook" above.

Build

See section "Build" above.

Deployment Plan

The purpose of this work product is to capture, in one document, the unique information that will be consumed by deployment engineers before and during the deployment to production of a particular release package. The deployment plan should contain the unique instructions for deploying a particular version of a product. By "unique instructions" we mean those things that are not part

Table 10 Develop solution increment—the task description

Activity	Task	Task description	Roles involved
Develop Solution Increment	Design the Solution	Identify the elements and devise the interactions, behavior, relations, and data necessary to realize some functionality. Render the design visually to aid in solving the problem and communicating the solution.	Developer (Primary), Architect (additional), Analyst (additional), Stakeholder (additional), Tester (additional)
Develop Solution Increment	Implement Developer Test	Implement one or more tests that enable the validation of the individual implementation elements through execution.	Developer (Primary), Tester (additional)
Develop Solution Increment	Implement Solution	The purpose of this task is to produce an implementation for part of the solution (such as a class or component), or to fix one or more defects. The result is typically new or modified source code, which is referred to the implementation.	Developer (Primary), Stakeholder (additional), Tester (additional)
Develop Solution Increment	Run Developer Tests	Run tests against the individual implementation elements to verify that their internal structures work as specified.	Developer (Primary)
Develop Solution Increment	Integrate and Create Build	This task describes how to integrate all changes made by developers into the code base and perform the minimal testing to validate the build.	Developer (Primary)

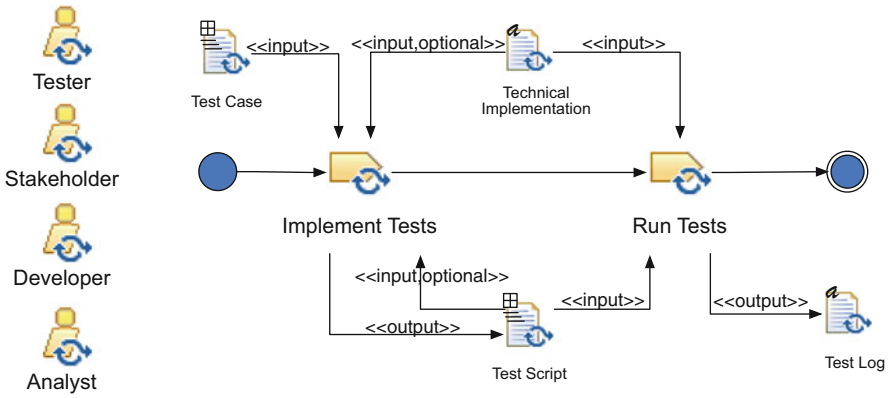

Fig. 33 The flow of tasks of the Test Solution activity

of a deployment engineer's normal procedures. Rather, they often are specific procedures and timing constraints that a deployment engineer should be aware of as they are rolling out a particular release. While a draft version of the deployment plan is typically developed by a development team, the deployment engineer is responsible for its contents and existence. A deployment plan normally consists of the following sections:

- The scope of the release and a general overview of the capabilities to be deployed
- The timing and dependencies for deploying components to various nodes
- The risks or issues associated with the release based on a risk assessment

Table 11 Test solution—the task description

Activity	Task	Task description	Roles involved
Test Solution	Implement Tests	To implement step-by-step Test Scripts that demonstrate the solution satisfies the requirements.	Tester (Primary), Analyst (Additional), Developer (Additional), Stakeholder (Additional)
Test Solution	Run Tests	Run the appropriate tests scripts, analyze results, articulate issues and communicate test results to the team.	Tester (Primary)

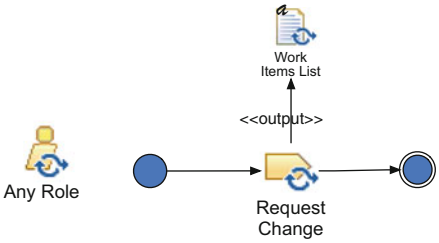

Fig. 34 The flow of tasks of the Ongoing Tasks activity

Table 12 Ongoing tasks—the task description

Activity	Task	Task description	Roles involved
Ongoing Tasks	Request Change	Capture and record change requests.	Any Role (perform)
Roles Identification	Design Scenarios	Each scenario in designed in form of sequence diagram thus depicting the details of agents interactions	System Analyst (perform), Domain Expert (assist)

- The customer organization, stakeholders, and end user community that will be impacted by the release
- The person or persons who have the authority to approve the release as "ready for production"
- The development team members responsible for delivering the release package to the Deployment Manager, along with contact information
- The approach for transitioning the release package to the Deployment Engineer, including appropriate communications protocols and escalation procedures
- The success criteria for this deployment; in other words, how will the Deployment Engineer know that the release is successful so it can report success

Design

See section "Design" above.

Developer Test

See section "Developer Test" above.

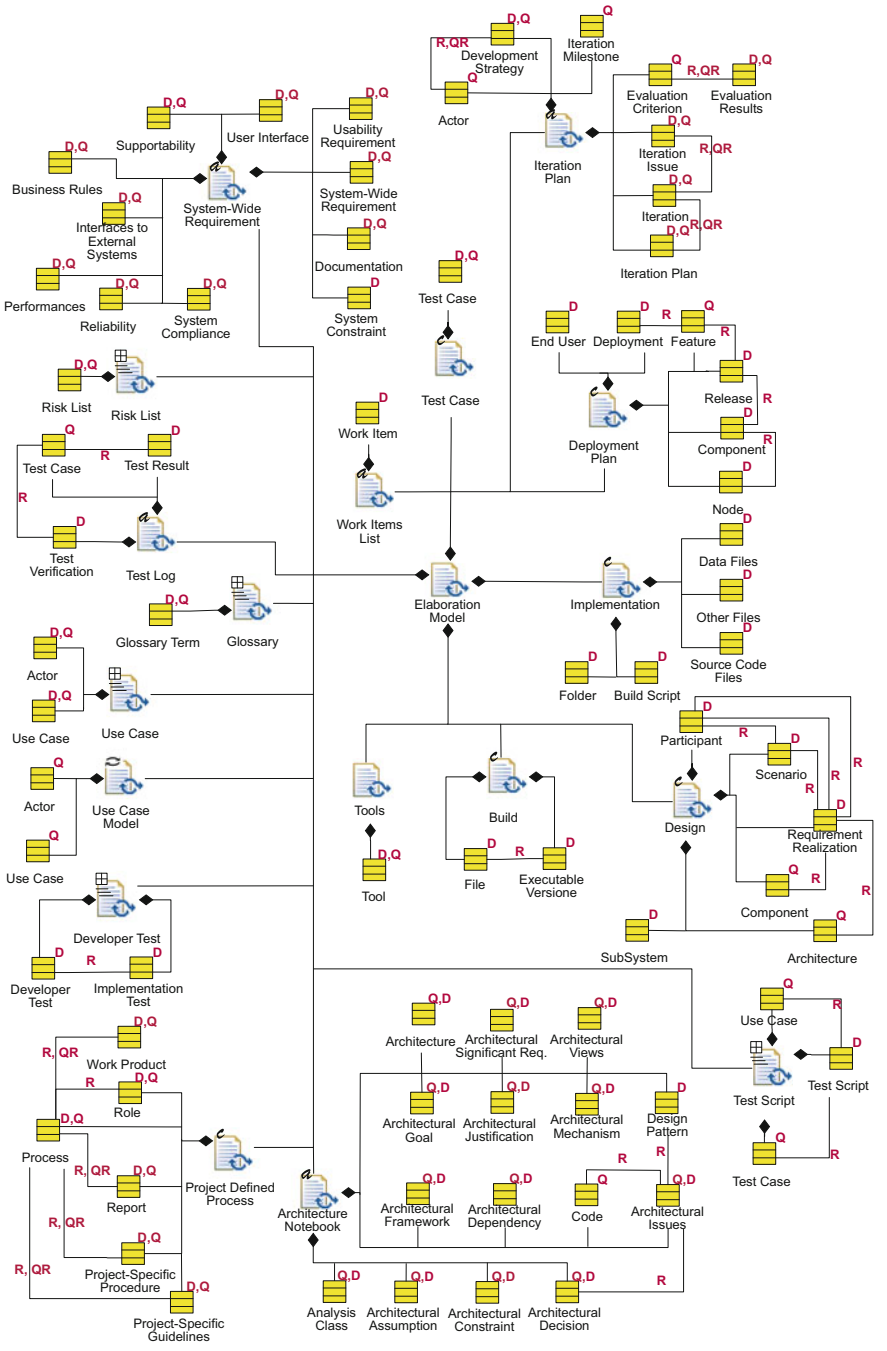

Fig. 35 The Elaboration phase documents structure

Table 13 Elaboration phase—work product kinds

Name	Description	Work product kind
Architecture Notebook	See sections "Architecture Notebook" and "Work Product Kinds".	
Build	See sections "Build" and "Work Product Kinds".	
Deployment Plan	A deployment plan is used to document all the information needed by deployment engineers to deploy a release successfully.	Composite
Design	See sections "Design" and "Work Product Kinds".	
Developer Test	See sections "Developer Test" and "Work Product Kinds".	
Glossary	See sections "Glossary" and "Work Product Kinds".	
Implementation	See sections "Implementation" and "Work Product Kinds".	
Iteration Plan	See sections "Iteration Plan" and "Work Product Kinds".	
Project Defined Process	See sections "Project Defined Process" and "Work Product Kinds".	
Project Plan	See sections "Project Plan" and "Work Product Kinds".	
Risk List	See sections "Risk List" and "Work Product Kinds".	
System-Wide Requirement	See sections "System-Wide Requirements" and "Work Product Kinds".	
Technical Architecture	See section "Work Product Kinds".	
Technical Design	This slot serves as an abstraction of high-level artifacts that describe the realization of required system functionality, and serves as an abstraction of the solution. Fulfilling Work Products: • Design	
Technical Implementation	This slot serves as an abstraction of high-level artifacts that describe the realization of required system functionality, and serves as an abstraction of the solution. Fulfilling Work Products: • Build • Implementation	
Technical Specification	See section "Work Product Kinds"	
Technical Test Results	See section "Work Product Kinds".	
Test Case	See sections "Test Case" and "Work Product Kinds".	
Test Log	See sections "Test Log" and "Work Product Kinds".	
Test Script	See sections "Test Script" and "Work Product Kinds".	
Tools	See sections "Tools" and "Work Product Kinds".	
Use Case	See sections "Use Case" and "Work Product Kinds".	
Use-Case Model	See sections "Use Case Model" and "Work Product Kinds".	
Vision	See sections "Vision" and "Work Product Kinds".	
Work Items List	See sections "Work Items List" and "Work Product Kinds".	

Glossary
See section "Glossary" above.

Implementation
See section "Implementation" above.

Iteration Plan
See section "Iteration Plan" above.

Project Defined Process
See section "Project Defined Process" above.

Release

The purpose of this work product is to

- Bring, at the team level, closure to a sprint/iteration or series of sprint/iterations by delivering working, tested software that can be potentially used by the end user community for whom the system was (or is being) developed.
- Deliver, at the program level, an integrated, value-added product increment to customers that was developed in parallel by multiple, coordinated, and synchronized development team members A release consists of integrated, compiled code that runs cleanly, independently, and in its entirety. This is an important rule because in order to be released or even "potentially shippable," a release increment must be able to stand on its own, otherwise it is not shippable. Releases can be created at either the program or team level.

  There are three potential uses for a release:
- It is not used outside of the program: It has been produced to minimize risks linked to technology and a program's capability to integrate components and to produce a Build. This situation usually happens at the beginning of a new product lifecycle.
- It is used by beta customers and the program manager: It allows end users to test it in a Beta test environment, which provides immediate feedback and reduces risks associated with user interface ergonomics. customer feedback will influence the program backlog for later consideration.
- It is deployed or shipped and used by end users: This result provides immediate value to the end users.

In many organizations, a program release typically is timeboxed to 2–3 months of development effort and 2–4 weeks of hardening which results in a scheduled release approximately every 90 days. Releases for individual development teams usually occur more often than those for programs, but there are no hard and fast rules regarding how often releases should be scheduled. The only requirement is that working software should be delivered "frequently" by both development teams and programs. Although the example timeframe described above is arbitrary, empirical evidence suggests it is about right for most companies and fits nicely into quarterly planning cycles. Each release has a set of release objectives and a projected delivery date; it also has a planned number of sprint/iterations.

Release Communications

The purpose of this work product is to inform all the various stakeholders that a release to production has taken place and the implemented features are now generally available. Sometimes, depending on the product user base, separate communiques might need to be prepared for each stakeholder group. In that case, this artifact should specify the different groups to which communications are directed, the method of communication (e.g., email, text or pager message, bulletin, newsletter, phone message, etc.). All communiques should be prepared in advance so that it is a matter of disseminating information when the release to production has been determined to be successful. Also included in this artifact is a listing of

the responsible parties who will execute the communications when a successful release has been declared (normally the Deployment Engineer), as well as the timing and dependencies of the communiques. While there is no prescribed format for the release communications artifact, each communique should indicate the preferred delivery mechanisms (e.g., beeper notification, telephone calls, a posting to an internal release website, live or pre-recorded presentations by senior management, etc.) and generally answer the following questions:

- Who are the parties (stakeholders) that are interested in knowing that a release to production has taken place?
- What specifically (features, functions, components) has been placed into production?
- Why is this release valuable to stakeholders and what business purpose does it serve?
- Where is the product available (on which platforms, geographical locations, business units, etc.)?
- How can the stakeholders access the system and under what circumstances?
- When was the product released (or when will it be released if the release date is in the future)?

Risk List
See section "Risk List" above.

System Wide Requirements
See section "System-Wide Requirements" above.

Test Case
See section "Test Case" above.

Test Log
See section "Test Log" above.

Test Script
See section "Test Script" above.

Tools
See section "Tools" above.

Use Case
See section "Use Case" above.

Use-Case Model
See section "Use Case Model" above.

Vision
See section "Vision" above.

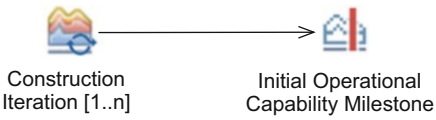

Fig. 36 The construction iteration inside the construction phase

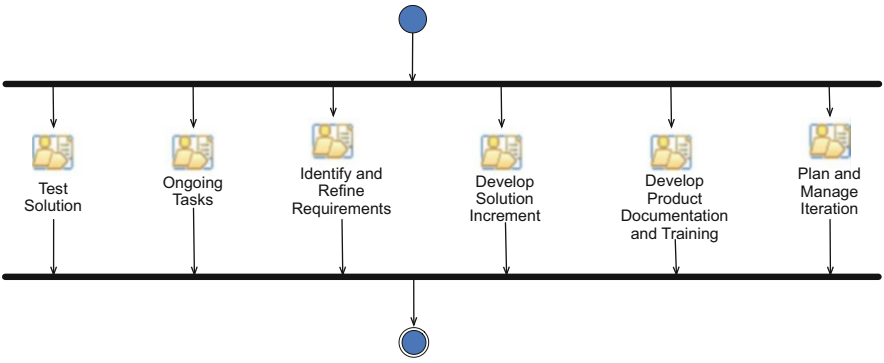

Fig. 37 The construction phase flow of activities

Work Items List

See section "Work Items List" above.

2.3 The Construction Phase

The Construction phase is composed by the Construction iteration as described in Fig. 36. The process flow within the iteration is detailed in Fig. 37.

The workflow inside each activity will be detailed in the following subsections (after the description of process roles). The Construction phase involves eight different process roles, five work products (four UML models and four text documents) and four guidance documents (one for each UML model) as described in Fig. 38. The phase is composed of six activities (i.e., Plan and Manage Iteration, Identify and Refine Requirements, Develop Solution Increment, Test Solution, Ongoing Tasks, Develop Product Documentation and Training), each of them composed of one or more tasks. Details of new activities are reported below (Figs. 39 and 40).

The description of the Identify and Refine Requirements activity in terms of tasks, roles, and work products is reported in Fig. 14.

The description of the Develop Solution Increment activity in terms of tasks, roles, and work products is reported in Fig. 27.

The description of the Test Solution activity in terms of tasks, roles, and work products is reported in Fig. 28. The description of the Ongoing Tasks activity in terms of tasks, roles, and work products is reported in Fig. 29.

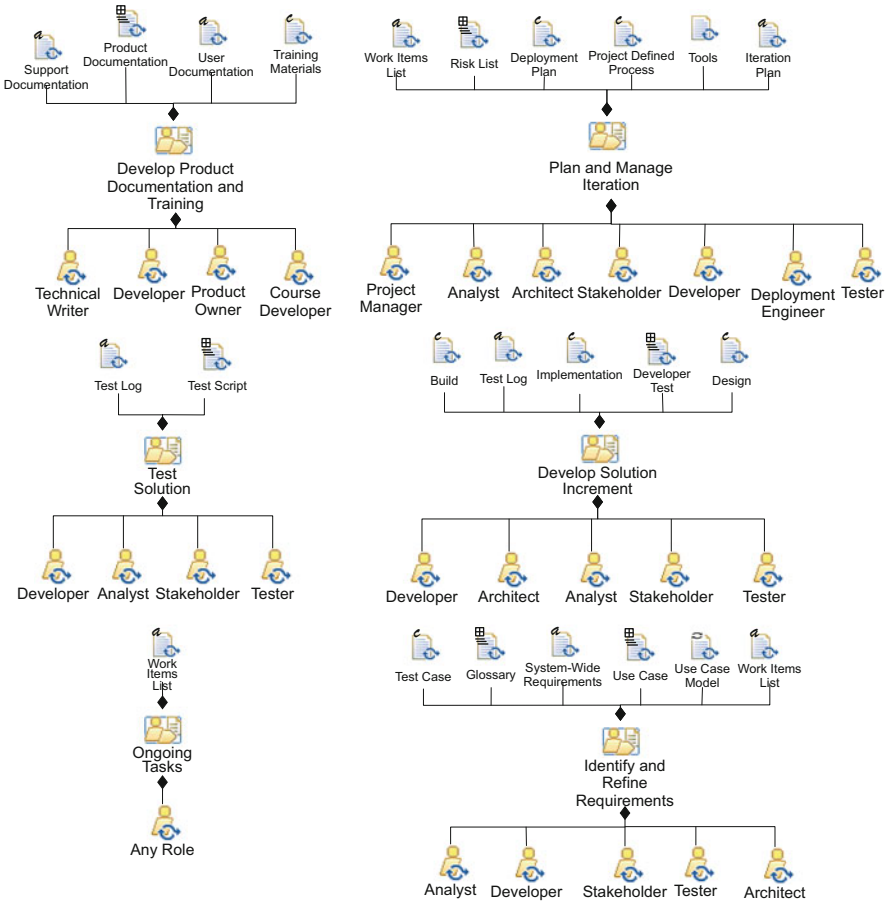

Fig. 38 The Construction phase described in terms of activities, output work products and involved stakeholders

2.3.1 Process Roles

Ten roles are involved in the Construction phase, which are described in the following subsections.

Analyst

She/he is responsible for the following tasks:

1. Detail System-Wide Requirements
2. Detail Use-Case Scenarios
3. Identify and Outline Requirements

She/he assists in the following tasks:

1. Assess Results
2. Create Test Cases

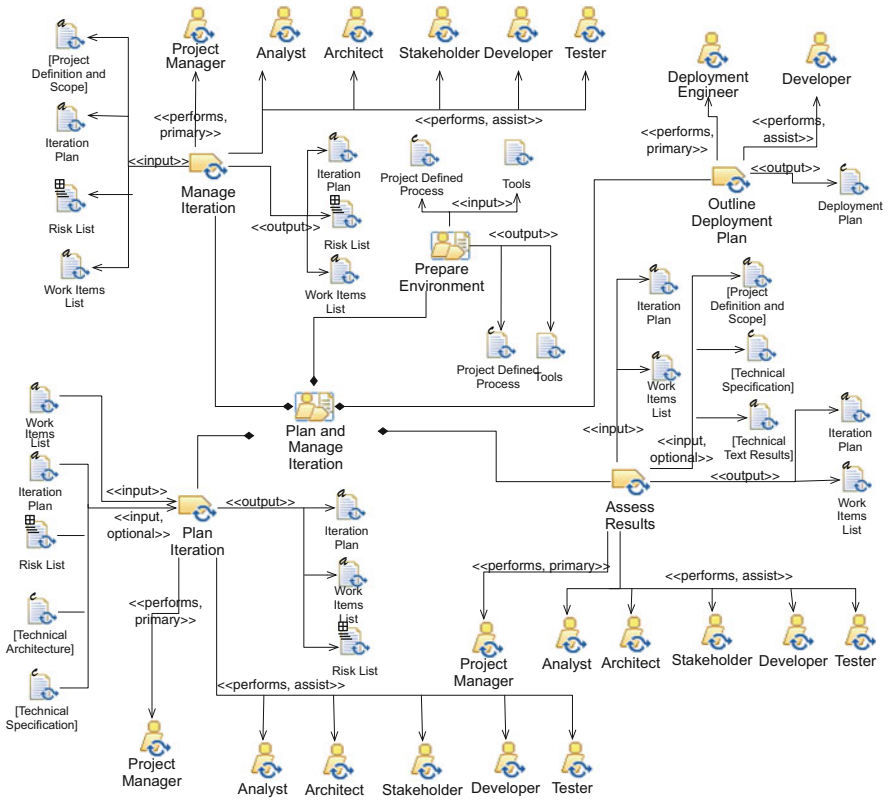

Fig. 39 The Plan and Manage Iteration activity described in terms of tasks, roles, and work products

3. Design the Solution
4. Implement Tests
5. Manage Iteration
6. Plan Iteration

Any Role

She/he is responsible for the following tasks:

1. Request Change

Architect

She/he is responsible for the following task:

1. Refine the Architecture

She/he assists in the following tasks:

1. Assess Results
2. Design the Solution

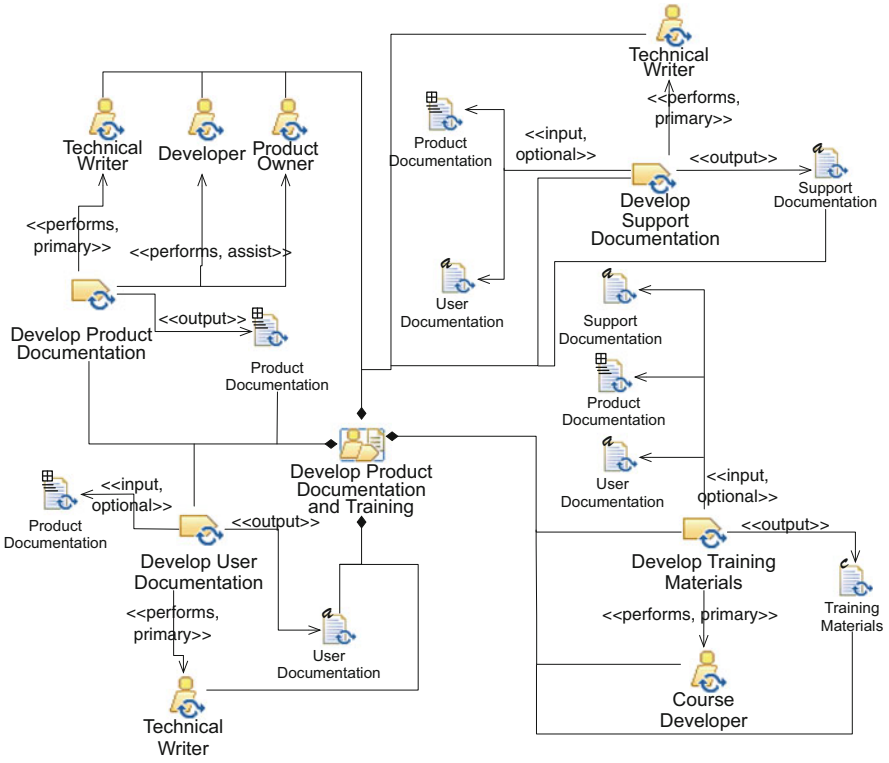

Fig. 40 The Develop Product Documentation and Training activity described in terms of tasks, roles, and work products

3. Detail System-Wide Requirements
4. Detail Use-Case Scenarios
5. Identify and Outline Requirements
6. Manage Iteration
7. Plan Iteration

Course Developer

She/he is responsible for the following tasks:

1. Develop Training Materials

Deployment Engineer

She/he is responsible for the following task:

1. Plan deployment

Developer

She/he is responsible for the following tasks:

1. Design the Solution
2. Implement Developer Tests
3. Implement Solution
4. Integrate and Create Build
5. Run Developer Tests

She/he assists in the following tasks:

1. Assess Results
2. Create Test Cases
3. Detail System-Wide Requirements
4. Detail Use-Case Scenarios
5. Develop Product Documentation
6. Identify and Outline Requirements
7. Implement Tests
8. Manage Iteration
9. Outline Deployment Plan
10. Plan Iteration

Process Engineer

She/he is responsible for the following tasks:

1. Deploy the process
2. Taylor the process

Product Owner

She/he assists in the following task:

1. Develop Product Documentation

Project Manager

She/he is responsible for the following task:

1. Assess Results
2. Manage Iteration
3. Plan Iteration

Stakeholder

She/he assists in the following tasks:

1. Assess Results
2. Create Test Cases
3. Design the Solution
4. Detail System-Wide Requirements
5. Detail Use-Case Scenarios
6. Identify and Outline Requirements
7. Implement Solution
8. Implement Tests
9. Manage Iteration
10. Plan Iteration

Technical Writer
She/he is responsible for the following task:
1. Develop Product Documentation
2. Develop Support Documentation
3. Develop User Documentation

Tester
She/he is responsible for the following tasks:
1. Create Test Cases
2. Implement Tests
3. Run Tests

She/he assists in the following tasks:
1. Assess Results
2. Design the Solution
3. Detail System-Wide Requirements
4. Detail Use-Case Scenarios
5. Identify and Outline Requirements
6. Implement Developer Tests
7. Implement Solution
8. Manage Iteration
9. Plan Iteration

Tool Specialist
She/he is responsible for the following tasks:
1. Set Up Tools
2. Verify Tool Configuration and Installation

2.3.2 Activity Details
The Construction phase includes six activities, which are described in the following subsections.

Plan and Manage Iteration
This activity is performed throughout the project lifecycle. The goal of this activity is to identify risks and issues early enough that they can be mitigated, to establish the goals for the iteration, and to support the development team in reaching these goals. The project manager and the team launch the iteration. The prioritization of work for a given iteration takes place. The project manager, stakeholders, and team members agree on what is supposed to be developed during that iteration. Team members sign up for the work items they will develop in that iteration. Each team member breaks down the work items into development tasks and estimates the effort. This provides a more accurate estimate of the amount of time that will be spent, and of what can be realistically achieved, in a given iteration. As the iteration runs, the team meets regularly to report status of work completed, the work to do next, and issues blocking the progress. In some projects, this status checking occurs in daily meetings, which allows for a more precise understanding of how

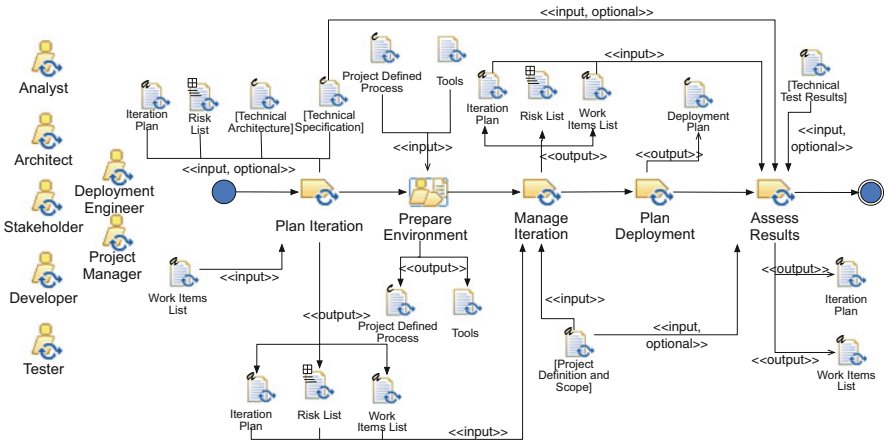

Fig. 41 The flow of tasks of the Plan and Manage Iteration activity

Table 14 Plan and manage iteration—the task description

Activity	Task	Task description	Roles involved
Plan and Manage Iteration	Plan Iteration	The purpose of this task is to identify the next increment of system capability, and create a fine-grained plan for achieving that capability within a single iteration.	Project Manager (perform), Analyst (assist), Architect (assist), Developer (assist), Stakeholder (assist), Tester (assist).
Plan and Manage Iteration	Manage Iteration	Help the team meet the iteration objectives and keep the project on track. Manage stakeholders' expectations as technical and practical discoveries are made during the project.	Project Manager (perform), Analyst (assist), Architect (assist), Developer (assist), Stakeholder (assist), Tester (assist). (perform), Domain Expert (assist)
Plan and Manage Iteration	Plan Deployment	If a deployment plan for the project already exists, update it to reflect the nature of this release. If this document does not exist, develop a deployment plan to indicate how this release will be rolled out to the production environment.	Deployment Engineer (perform), Developer (assist).
Plan and Manage Iteration	Assess Results	Demonstrate the value of the solution increment that was built during the iteration and apply the lessons learned to modify the project or improve the process.	Project Manager (perform), Analyst (assist), Architect (assist), Developer (assist), Stakeholder (assist), Tester (assist).

the work in an iteration is progressing. As necessary, the team makes corrections to achieve what was planned. The overall idea is that risks and issues are identified and managed throughout the iteration, and everyone knows the project status in a timely manner. During iteration assessments, the key success criterion is the demonstration that planned functionality has been implemented. Lessons learned are captured in order to modify the project or improve the process. If the iteration end coincides with the phase end, make sure the objectives for that phase have been met. The flow of tasks inside this activity is reported in Fig. 41, and the tasks are detailed in Table 14.

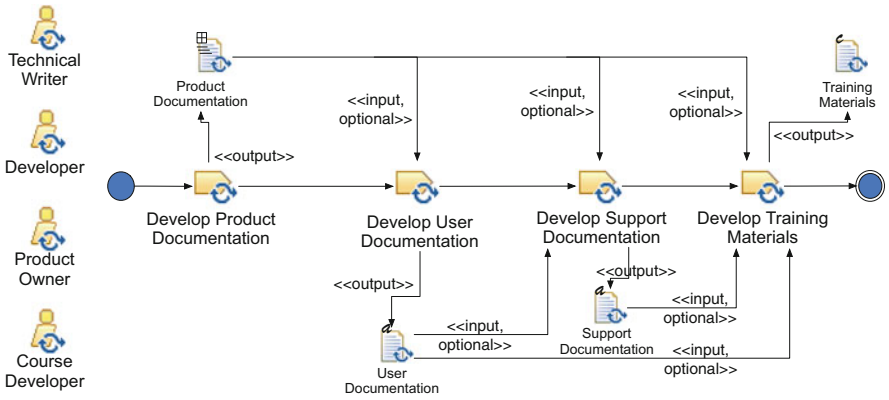

Fig. 42 The flow of tasks of the Develop Product Documentation and Training activity

Prepare Environment

See section "Prepare Environment" above.

Identify and Refine Requirements

See section "Identify and Refine Requirements" above.

Develop Solution Increment

See section "Develop Solution Increment" above.

Test Solution

See section "Test Solution" above.

Ongoing Tasks

See section "Ongoing Tasks" above.

Develop Product Documentation and Training

The goal of this activity is to prepare product documentation and training materials. The flow of tasks inside this activity is reported in Fig. 42, and the tasks are detailed in Table 15.

2.3.3 Work Products

The Construction phase generates four work products. Their relationships with the system meta-model elements are described in Fig. 43.

WorkProduct Kinds

Table 16 describes the work products of the Construction phase according to their kinds.

Table 15 Develop product documentation and training—the task description

Activity	Task	Task Description	Roles Involved
Develop Product Documentation and Training	Develop Product Documentation	The purpose of this task is to document enough information about the features that were developed in a particular release to be useful to customers throughout the life of the product.	Technical Writer (perform), Developer (assist), Product Owner (assist).
Develop Product Documentation and Training	Develop User Documentation	The purpose of this task is to provide useful information to end users of the product being released into production.t.	Technical Writer (perform), Developer (assist), Product Owner (assist).
Develop Product Documentation and Training	Develop Support Documentation	The purpose of this task is to ensure that the personnel who are tasked with supporting the system have enough information about the product to perform their jobs effectively after the product has been placed into production.	Technical Writer (perform)
Develop Product Documentation and Training	Develop training Materials	The purpose of this task is to enable adoption of the product and to encourage its proper use.	Course Developer (perform)

Product Documentation

Product documentation is created for the benefit of the marketing arm of an organization, the program manager, and those people who must assess the business value of a particular system. This is an often overlooked aspect of development team implementation. The team should consider that the customers of a particular release usually are not technical themselves, but do require a detailed enough understanding of how their product operates and how it meets stated business goals and needs

User Documentation

User documentation might include all or parts of user manuals (electronic or paper-based), tutorials, frequently asked questions (FAQs), on-line Help Files, installation instructions, work instructions, operational procedures, etc.

Support Documentation

Support documentation usually is developed for the three common tiers of a support organization. Tier 1 typically is the Help Desk where users call when they have a problem with a particular system. Tier 1 support personnel normally answer basic questions and, if necessary, log a ticket and escalate it to the appropriate Level 2 support desk.

Tier 2 support personnel may deal with more complex questions or issues regarding an application and might need to do some research on the characteristics of the system to provide an answer. If that person cannot resolve the issue, the ticket is escalated to Tier 3 support personnel who have a deeper understanding of the application's code and the technology support the system's architecture.

To properly convey the necessary information to each support tier, the application's code base should be well commented and logically organized. This approach will facilitate the development of the support documentation. Support documentation typically includes

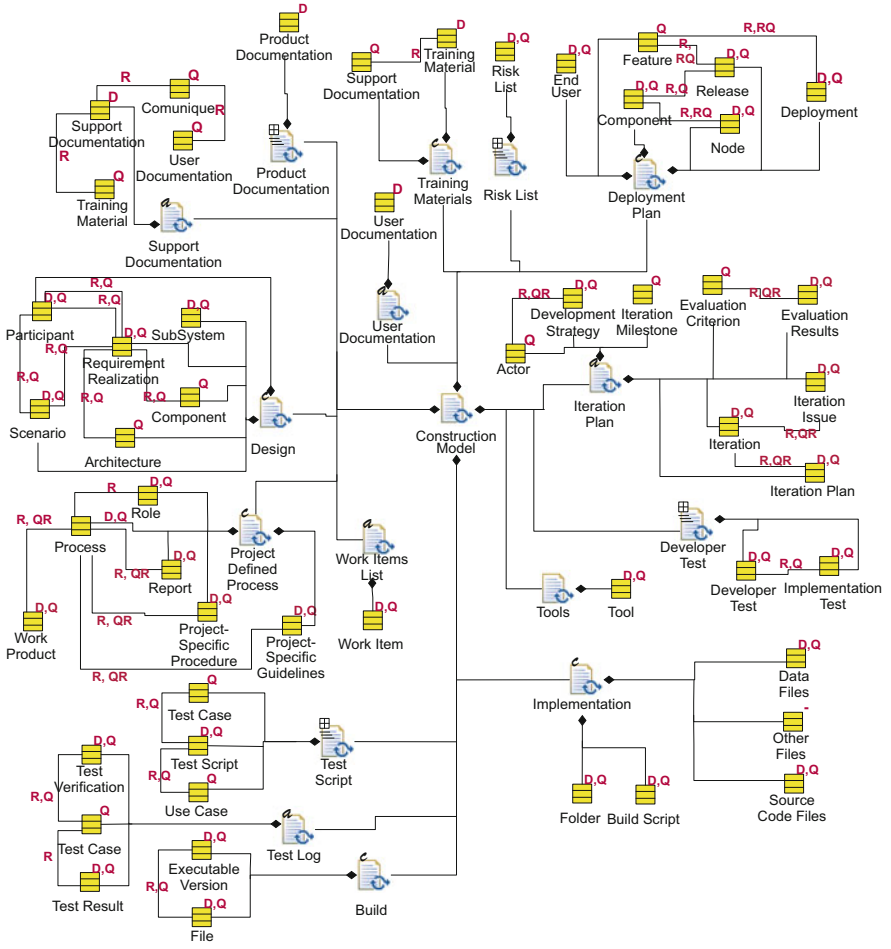

Fig. 43 The Construction phase documents structure

- User manuals with work instructions, process descriptions, and procedures
- Communications, training, and knowledge transfer deliverables
- Support and operations manuals
- Service information, including Help Desk scripts

Training Materials

Training materials that can be used to train end users and production support personnel might consist of

- Presentation slides
- Handouts
- Job aids
- Tutorials

Table 16 Construction phase—work product kinds

Name	Description	Work product kind
Architecture Notebook	See sections "Architecture Notebook" and "Work Product Kinds".	
Build	See sections "Build" and "Work Product Kinds".	
Deployment Plan	See sections "Deployment Plan" and "WorkProduct Kinds".	Composite
Design	See sections "Design" and "Work Product Kinds".	
Developer Test	See sections "Developer Test" and "Work Product Kinds".	
Glossary	See sections "Glossary" and "Work Product Kinds".	
Implementation	See sections "Implementation" and "Work Product Kinds".	
Iteration Plan	See sections "Iteration Plan" and "Work Product Kinds".	
Product Documentation	Information about a specific product that has been captured in an organized format.	Structured
Project Defined Process	See sections "Project Defined Process" and "Work Product Kinds".	
Project Plan	See sections "Project Plan" and "Work Product Kinds".	
Risk List	See sections "Risk List" and "Work Product Kinds".	
Support Documentation	Documents used by members of a production support team that provide information about how to service and support a specific product.	Free Text
System-Wide Requirement	See sections "System-Wide Requirements" and "Work Product Kinds".	
Test Case	See sections "Test Case" and "Work Product Kinds".	
Test Log	See sections "Test Log" and "Work Product Kinds".	
Test Script	See sections "Test Script" and "Work Product Kinds".	
Tools	See sections "Tools" and "Work Product Kinds".	
Training Materials	This work product represents all the materials needed to train end users and production support personnel on the features and inner workings of a product for a particular release.	Composite.
Use Case	See sections "Use Case" and "Work Product Kinds".	
Use-Case Model	See sections "Use Case Model" and "Work Product Kinds".	
User Documentation	Documents that can be utilized by the end users of a particular system or product. This type of documentation typically is written in a way that enables system users to easily find information they need to use the product.	Free Text.
Vision	See sections "Vision" and "Work Product Kinds".	
Work Items List	See sections "Work Items List" and "Work Product Kinds".	

- On-line demos
- Video vignettes
- Lab exercises
- Quizzes
- Workshop materials, etc.

2.4 The Transition Phase

The Transition phase is composed by the Inception iteration as described in Fig. 44. The process flow within the iteration is detailed in Fig. 45.

Fig. 44 The Transition iteration inside the Transition phase

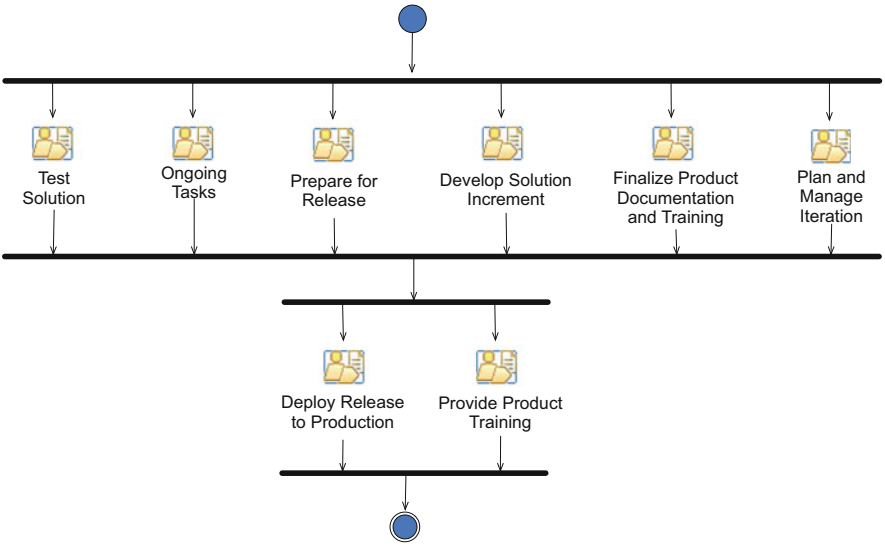

Fig. 45 The Transition phase flow of activities

The workflow inside each activity will be detailed in the following subsections (after the description of process roles). The Transition phase involves thirteen different process roles, seventeen work products as described in Fig. 46.

The phase is composed of eight activities each of them composed of one or more tasks as described in the following. Activities are Plan and Manage Iteration, Develop Solution Increment, Test Solution, Finalize Product Documentation and Training, Prepare for Release, Package the Release, Ongoing Tasks, Provide Product Training. The description of the Plan and Manage Iteration activity, Develop Solution Increment activity and Test Solution activity in terms of tasks, roles, and work products can be found respectively in Figs. 12, 27, and 28.

Details of new activities are reported in Figs. 47, 48, 49, and 50.

2.4.1 Process Roles

Analyst

She/he assists in the following tasks:

1. Assess Results
2. Design the Solution

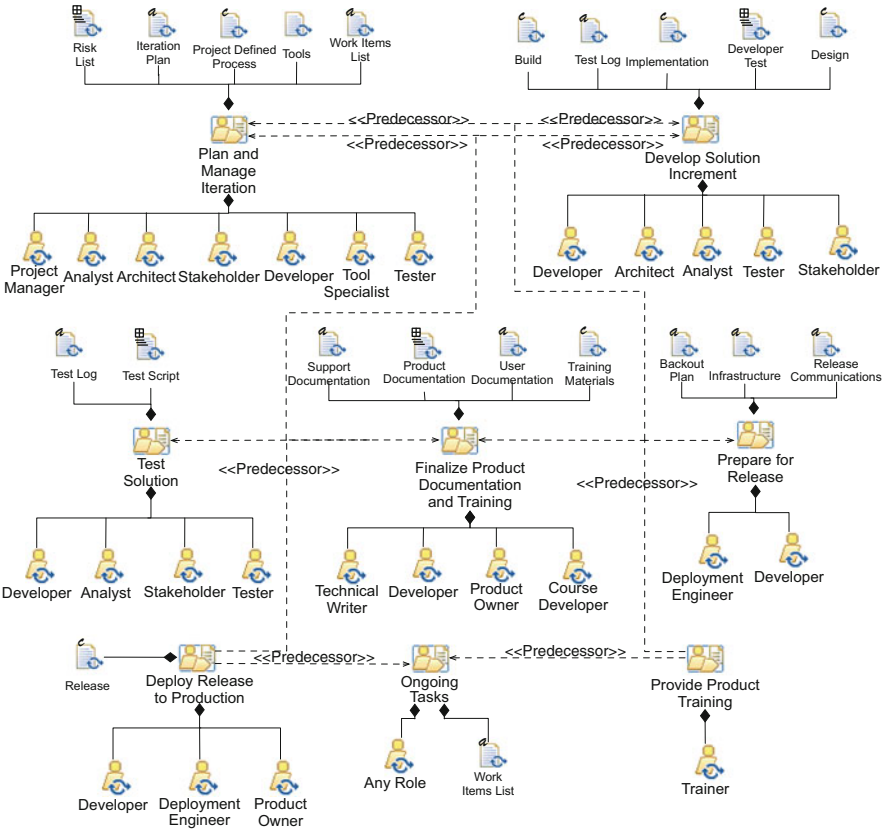

Fig. 46 The Transition phase described in terms of activities, output work products and involved

3. Implement Tests
4. Manage Iteration
5. Plan Iteration

Any Role

She/he is responsible for the following tasks:

1. Request Change

Architect

She/he assists in the following tasks:

1. Assess Results
2. Design the Solution
3. Manage Iteration
4. Plan Iteration

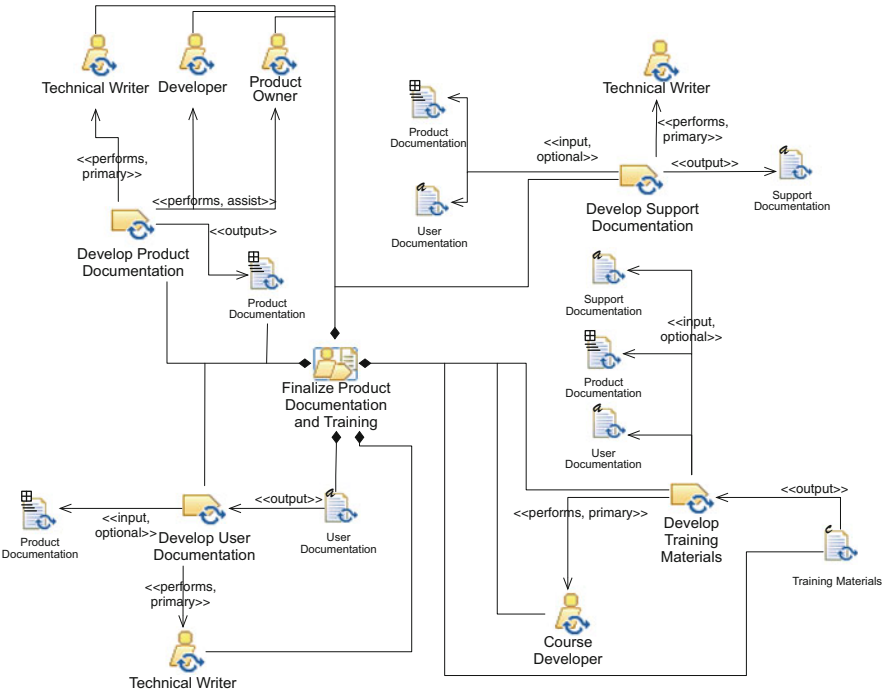

Fig. 47 The Finalize Product Documentation and Training activity described in terms of tasks, roles, and work products

Course Developer

She/he is responsible for the following tasks:

1. Develop Training Materials

Deployment Engineer

She/he is responsible for the following tasks:

1. Deliver Release Communications
2. Develop Release Communications
3. Execute Backout Plan (if necessary)
4. Execute Deployment Plan
5. Install and Validate Infrastructure
6. Verify Successful Deployment
7. Develop Backout Plan

She/he assists in the following tasks:

1. Package the Release

Developer

She/he is responsible for the following tasks:

1. Design the Solution
2. Develop Backout Plan

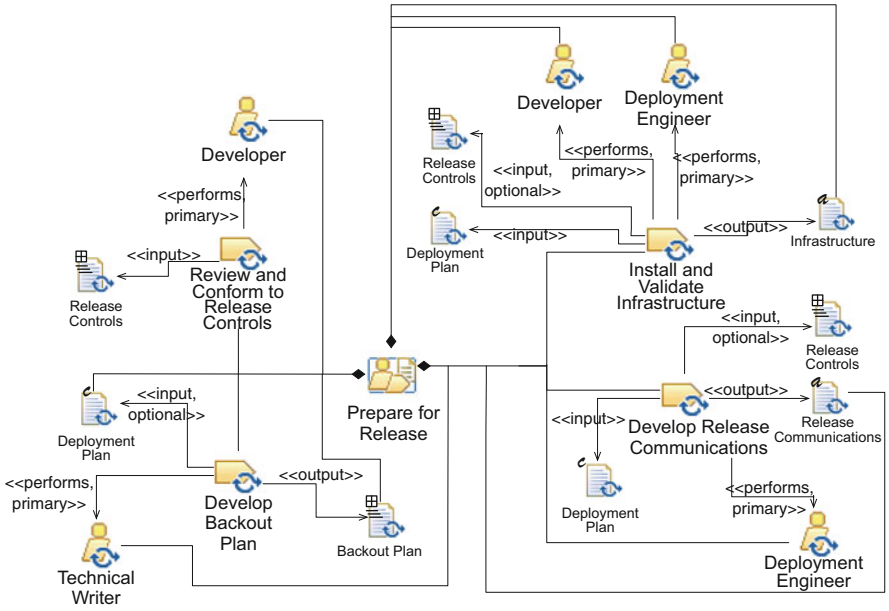

Fig. 48 The Prepare for Release activity described in terms of tasks, roles, and work products

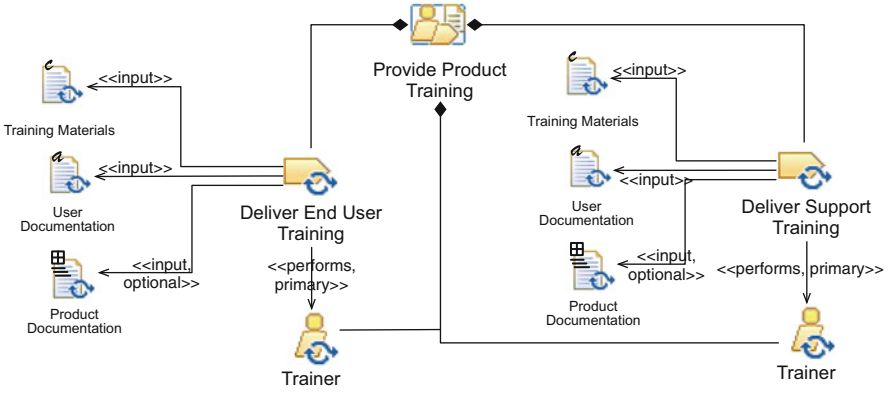

Fig. 49 The Provide Product Training activity described in terms of tasks, roles, and work products

3. Implement Developer Tests
4. Implement Solution
5. Install and Validate Infrastructure
6. Integrate and Create Build
7. Package the Release
8. Review and Conform to Release Controls
9. Run Developer Tests

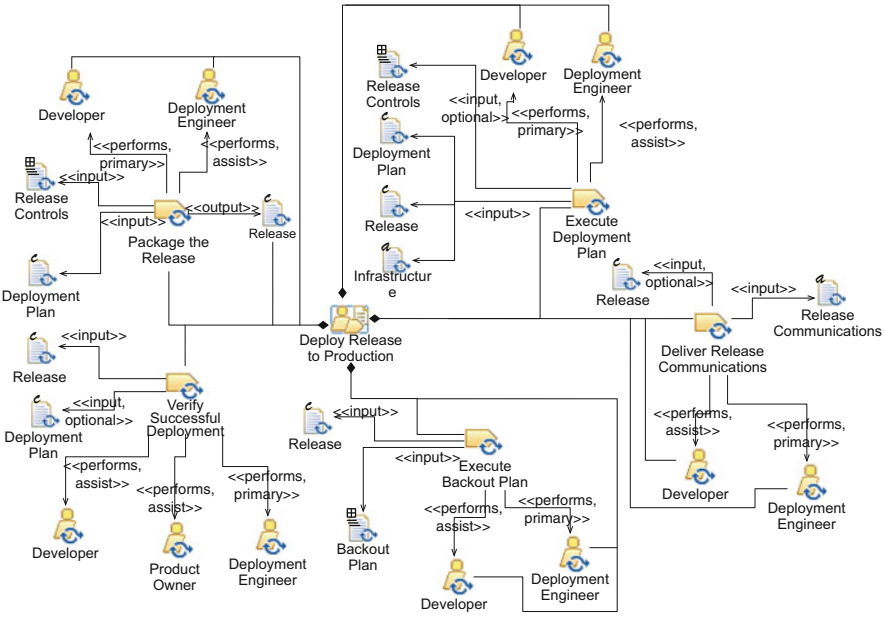

Fig. 50 The Deploy Release to Production activity described in terms of tasks, roles, and work products

She/he assists in the following tasks:

1. Assess Results
2. Deliver Release Communications
3. Develop Product Documentation
4. Execute Backout Plan (if necessary)
5. Execute Deployment Plan
6. Implement Tests
7. Manage Iteration
8. Outline Deployment Plan
9. Plan Iteration
10. Verify Successful Deployment

Process Engineer

She/he is responsible for the following tasks:

1. Deploy the Process
2. Tailor the Process

Product Owner

She/he assists in the following tasks:

1. Develop Product Documentation
2. Verify Successful Deployment

Project Manager
She/he is responsible for the following tasks:
1. Assess Results
2. Manage Iteration
3. Plan Iteration

Stakeholder
She/he assists in the following tasks:
1. Assess Results
2. Design the Solution
3. Implement Solution
4. Implement Tests
5. Manage Iteration
6. Plan Iteration

Technical Writer
She/he is responsible for the following tasks:
1. Develop Product Documentation
2. Develop Support Documentation
3. Develop User Documentation

Tester
She/he is responsible for the following tasks:
1. Implement Tests
2. Run Tests

She/he assists in the following tasks:
1. Assess Results
2. Design the Solution
3. Implement Developer Tests
4. Implement Solution
5. Manage Iteration
6. Plan Iteration

Tool Specialist
She/he is responsible for the following tasks:
1. Set Up Tools
2. Verify Tool Configuration and Installation

Trainer
She/he is responsible for the following tasks:
1. Deliver End User Training
2. Deliver Support Training

2.4.2 Activity Details
The Transition phase includes eight activities, which are described in the following subsections.

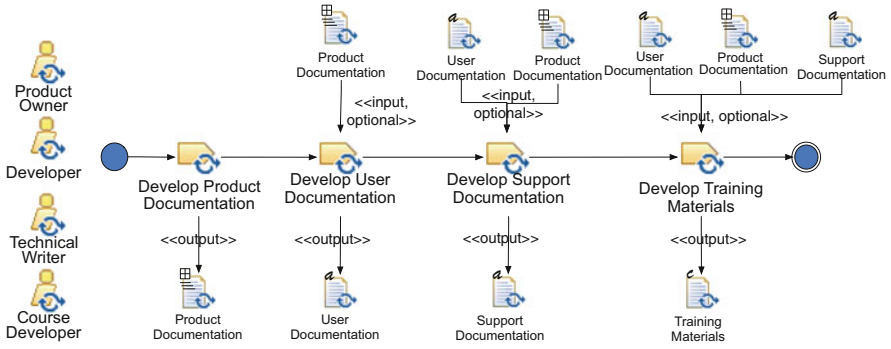

Fig. 51 The flow of tasks of the Finalize Product Documentation and Training activity

Plan and Manage Iteration

See section "Plan and Manage Iteration" above.

Develop Solution Increment

See section "Develop Solution Increment" above.

Test Solution

See section "Test Solution" above.

Finalize Product Documentation

This activity prepares product documentation and training materials. The flow of tasks inside this activity is reported in Fig. 51, and the tasks are detailed in Table 17.

Prepare for Release

This activity prepares product documentation and training materials. The flow of tasks inside this activity is reported in Fig. 52, and the tasks are detailed in Table 18.

Provide Product Training

This activity provides product training. The flow of tasks inside this activity is reported in Fig. 53, and the tasks are detailed in Table 19.

Ongoing Tasks

See section "Ongoing Tasks" above.

Deploy Release to Production

This activity results in the release of a set of integrated components into the production environment. The flow of tasks inside this activity is reported in Fig. 54, and the tasks are detailed in Table 20.

Table 17 Finalize product documentation and training—the task description

Activity	Task	Task description	Roles involved
Finalize Product Documentation	Develop Product Documentation	Development team members sometimes take documentation for granted, or do not give it enough consideration. However, after a product is delivered, customers who pay for the system and for support often do not have enough information to effectively manage the product. If a technical writer is made available to a development team, that role often takes the burden off the team for developing the formal product documentation and for ensuring that it is in the correct format and business language. If a technical writer is not available, the development team and product owner must make every effort to create enough documentation to ensure that the features that have been developed for each release are understood and can be communicated effectively by the paying customer to their stakeholders. Delivering a professionally developed product requires that a development team provide the customer with accurate, detailed, and comprehensive product documentation. This task includes the following steps: 1. Identify features of current release 2. Document each feature 3. Review product documentation with stakeholders 4. Update product documentation as necessary 5. Deliver product documentation	Technical Writer (perform), Developer (assist), Product Owner (assist).
Finalize Product Documentation	Develop User Documentation	User documentation might include all or parts of user manuals (electronic or paper-based), tutorials, frequently asked questions (FAQs), on-line Help Files, installation instructions, operational procedures, etc. User documentation often is used as the basis for training materials - if the documentation is of poor quality, the training materials might not be any better. Without good user documentation, a system might be well developed by a development team but might not meet the End User's expectations because they will not be able operate the application effectively. This task includes the following steps: 1. Determine user documentation contents 2. Leverage product documentation 3. Leverage other materials 4. Write user documentation content 5. Perform quality review 6. Deliver user documentation	Technical Writer (perform)
Finalize Product Documentation	Develop Support Documentation	Support documentation often is the most overlooked aspect of a documentation effort. Anyone who has had the opportunity to provide end user support for a particular application can appreciate how important effective, well-written support documentation can be. This documentation very often is technical in nature and differs significantly from user or product documentation, which normally is written for the lay person. The development team should do its best to make sure that personnel who perform an IT support role have the right amount and the relevant type of information necessary to support the application, whether they provide Tier 1, Tier 2, or Tier 3 support. Support documentation often is developed based on these three different support categories. How effectively the code base	Technical Writer (perform)

(continued)

Table 17 (continued)

Activity	Task	Task description	Roles involved
		is commented and the ease with which those comments are found and understood contributes to the quality and quantity of support documentation. This task includes the following steps: 1. Determine support documentation contents 2. Leverage available materials 3. Write support documentation 4. Perform quality review 5. Deliver support documentation	
Finalize Product Documentation	Develop Training Materials	Having the correct amount and type of materials available to adequately train end users and supporters of an application is necessary to promote usability and to achieve the desired business value. If a course developer is available, they can assume most of the burden of creating the training materials, lab exercises, and workshops for delivery of those courses to either end users or support personnel. If a course developer is not available, development team members should take the time to properly develop a suite of training materials for the feature set developed during a release. Although different parts of training materials should be created during feature development sprint/iterations, the bulk of the work (and the integration of the materials) usually is reserved for the release sprint/iteration that occurs immediately before a scheduled release. This task includes the following steps: 1. Determine scope of training materials 2. Develop user training materials 3. Develop support training materials 4. Perform quality review 5. Perform dry run 6. Deliver training materials	Course Developer (perform)

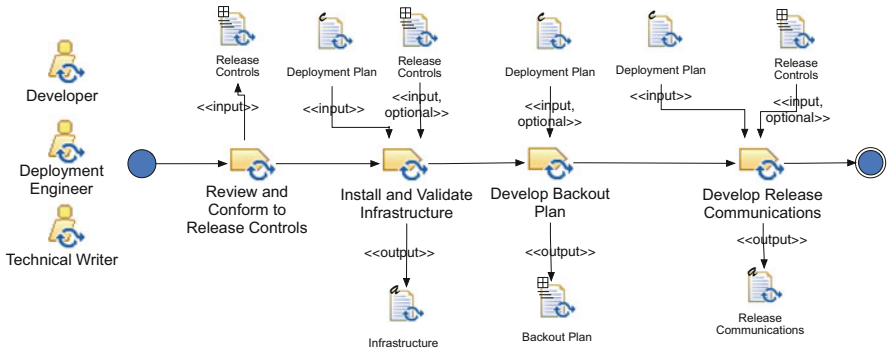

Fig. 52 The flow of tasks of the Prepare for Release activity

2.4.3 Work Products

The Transition phase generates nine work products. Their relationships with the system meta-model elements are described in Fig. 55.

Table 18 Prepare for release—the task description

Activity	Task	Task description	Roles involved
Prepare for Release	Review and Conform to Release Controls	Release controls describe the minimum number of requirements that a software package must adhere to before being released into production. This is especially important if a development team is new or emerging, because they might not be aware of the great responsibilities a deployment manager has. In fact, a deployment manager is responsible to senior management for ensuring that nothing is placed into production that does not conform to the rigid controls designed to protect the IT organization's ability to successfully deliver IT services to internal and external customers. Release controls typically consist of • Release or deployment plan • Backout plan • Release component definitions • Release package integrity verification • References to configuration items (CIs) • Customer approval • Ready for transfer to operations and support staff This task includes the following steps: 1. Locate release controls 2. Review release controls 3. Ensure the team release conforms to the controls	Technical Writer (perform), Developer (perform)
Prepare for Release	Install and Validate Infrastructure	A release package cannot be deployed to production if the environmental infrastructure within which the release will be run is not sufficiently built or tested. Whether the release is deployed as a "push" (where the application is deployed from a central point and proactively delivered to target locations) or a "pull" (where the application is made available at central point and pulled by a user at a time of their choosing), the infrastructure needed to support the application must be considered and implemented. Some key aspects of installing and/or validating the desired infrastructure: • Identify the requirements and components of the environment configuration • Determine the lead times required to establish the infrastructure environments • Procure and install the infrastructure components that are not yet available • Test the newly installed infrastructure components • Test the integration of newly installed components with the rest of the environmental configuration • Validate other aspects of the infrastructure including – Security components and their integration – Database connectivity and security – License management, as appropriate – Configuration management, in terms of configuration items (CIs) This task includes the following steps: 1. Identify infrastructure needs 2. Procure components 3. Schedule components for installation 4. Install and test components 5. Validate other component aspects	Deployment Engineer, Developer (perform)

(continued)

Table 18 (continued)

Activity	Task	Task description	Roles involved
Prepare for Release	Develop Backout Plan	A rollback might be needed for a variety of reasons, including corruption of the production code base, inoperable components, an unplanned undesirable effect of the release on other production systems, an unhappy customer, etc. The Development team should provide the production support organization with a specific plan and decision criteria made available to them to avoid or minimize service interruptions. This task includes the following steps: 1. Determine if backout plan exists 2. Develop the backout plan (if applicable) 3. Update the backout plan (if applicable)	Developer (perform), Deployment Engineer (assist)
Prepare for Release	Develop Release Communications	When a release is pushed to production, all the stakeholders of that product should be notified that the event has happened and what the release means to each of the stakeholders. Often, the output of this task does not need to be created from scratch; for products that plan multiple releases, just updating the communique details for each release might be enough. However, if any of the stakeholder groups change, or there is a significant difference in the product distribution, more significant content might need to be developed. A development team can develop high quality software, but if messaging to the stakeholders is conducted poorly or not at all, the end user experience might be degraded. By simply answering the questions "who, what, when, where, why, and how" in a format appropriate for each stakeholder group, a product release can become a more satisfying experience for all those involved. This task includes the following steps: 1. Identify stakeholders for this release 2. Draft communique for each stakeholder group 3. Provide communiques to deployment manager	Deployment Engineer (perform)

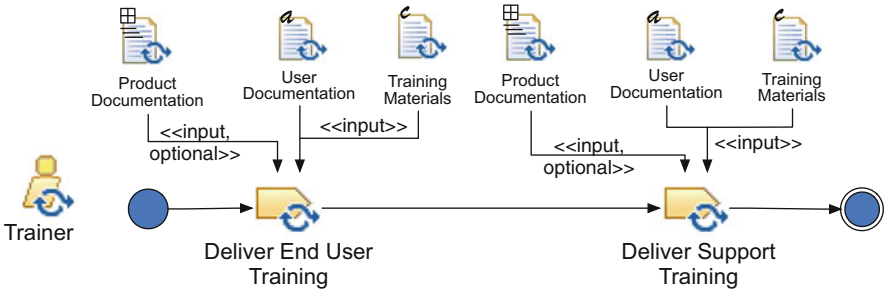

Fig. 53 The flow of tasks of the Provide Product Training activity

WorkProduct Kinds

Table 21 describes the work products of the Construction phase according to their kinds.

Table 19 Provide product training—the task description

Activity	Task	Task description	Roles involved
Provide Product Training	Deliver end user Training	Often, a trainer will deliver training to end users; rarely will the development team deliver training because they are busy developing additional features for this or other systems. If a trainer is not available, the product owner might have to train the end users. End user training usually consists of presentation slides, job aids, hands-on labs and exercises, or workshops that integrate these methods into an environment that the end users can understand and relate to. This task includes the following steps: 1. Validate user training logistics 2. Prepare for user training delivery 3. Deliver user training and gather feedback 4. Provide feedback to the program level	Technical Writer (perform)
Provide Product Training	Deliver Support Training	Because a release ultimately will have to be supported by Help Desk or other technical personnel, a set of training materials designed to convey key concepts to those individuals must be developed and delivered by the development team or by a technically oriented trainer. Delivery of this training might include presentation slides, job aids, lab exercises, or workshops designed to provide real-world scenarios to accelerate learning and retention. In addition, the training materials might be different for each level (tier) of support. This task includes the following steps: 1. Validate support training logistics 2. Prepare for support training delivery 3. deliver support training and gather feedback 4. Provide feedback to the program	Trainer (perform)

Backout Plan

The purpose of this work product is for the development team to provide, in one document, all the information needed by the production support organization to determine if a rollback is needed, who will authorize it, how it will be performed, etc. While someone on the development team normally authors a draft version of the Backout Plan, the Deployment Engineer is ultimately responsible for its contents and existence. A backout plan typically answers the following questions:

- Under what circumstances will a rollback be required? Or conversely, under what circumstances will the deployment be considered a success?
- What is the time period within which a rollback can take place?
- Which authorizing agent will make the decision to revert?
- Who will perform the rollback and how soon after the decision has been made will the rollback be performed?
- What procedures (manual and automated) will be followed to execute the rollback?
- What other contingency measures or available workarounds should be considered?

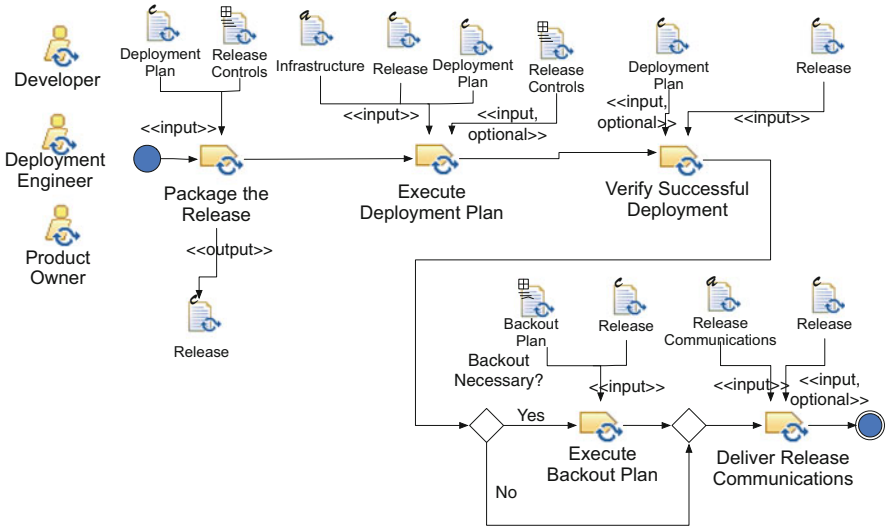

Fig. 54 The flow of tasks of the Deploy Release to Production activity

- What is the expected time required to perform a reversion?
- What are the communication procedures required in the event of a backout?
- Has the Backout Plan been successfully tested?

Infrastructure

Infrastructure normally is defined as anything that supports the flow and processing of information in an organization. The infrastructure needed to support a release package normally includes

- Software, including
  - Operating systems and applications for servers and clients
  - Desktop applications
  - Middleware
  - Protocols
- Hardware
- Networks, including
  - Routers
  - Aggregators
  - Repeaters
  - Other transmission media devices that control movement of data and signals
- Facilities

Table 20 Deploy release to production—the task description

Activity	Task	Task description	Roles involved
Deploy Release to Production	Package the Release	The key activities normally used to package a release: • Assemble the components and integrate them through a normal (i.e., continuous integration) or release build script • Install the release package in one or more test environments and verify its integrity • Tag the elements of the release package in the code base to create a baseline • Package appropriate documentation to accompany the release: – Deployment plan – Build plan, procedures, and scripts – Backout plan – Relevant licensing information – Relevant infrastructure information – Release communiques This task includes the following steps: 1. Assemble components 2. Test the release 3. Tag source code repository 4. Package release documentation 5. Deliver release package	Developer (perform), Deployment Engineer (assist)
Deploy Release to Production	Execute Deployment Plan	This task is straightforward: follow the procedures in the Deployment Plan for the rollout of a specific product release. If the deployment plan does not exist or it is poorly constructed, this task might be much more difficult. The main point here is that to achieve a high probability of success, the development team should have previously developed a detailed plan that organizes and articulates all the unique instructions for deploying that particular release. Because an experienced deployment engineer normally executes this task, they might be able to overcome any missing deployment procedures or content. However, that is not an excuse for a development team to not develop the plan's contents. This task includes the following steps: 1. Review deployment plan 2. Release code	Deployment Engineer (perform), Developer (assist)
Deploy Release to Production	Verify Successful Deployment	Using the success criteria documented either in the deployment plan or in the backout plan, the deployment engineer, in collaboration with the development team, will determine whether the rollout can be declared a success or not. If the deployment is successful, the previously prepared release communiques should be delivered. If the deployment is unsuccessful, then the backout plan should be invoked. This task includes the following steps: 1. Test production release 2. Run manual tests 3. Determine if release should be reversed	Deployment Engineer (perform), Developer, Product Owner (assist)
Deploy Release to Production	Execute Backout Plan (if necessary)	Assuming a backout plan is available for this release, the deployment engineer (or development team) will follow the instructions for reversing the installation of the product into production, if there is a problem. While the plan might have been written with good intentions, sometimes key procedures are	Deployment Engineer (perform), Developer (assist)

(continued)

Table 20 (continued)

Activity	Task	Task description	Roles involved
		missing or have not been thought out. The team backing out the release should be aware that blindly following the backout plan might not be the best approach. It is best to consider the unique circumstances within which the deployment has failed and rely on common sense and experience when executing the backout plan. This task includes the following steps: 1. Identify release problem(s) 2. Backout the release 3. Determine if the backout was successful 4. Communicate the backout	
Deploy Release to Production	Deliver Release Communications	This task represents the distribution of communiques that were prepared beforehand as part of the release communications artifact. Although the development team is responsible for preparing the communications, the responsibility for sending the communiques normally is assigned to the deployment engineer, if that is the organizational protocol. This task includes the following steps: 1. Validate the communiques 2. Send release communications 3. Validate communications were received	Deployment Engineer (perform), Developer (assist)

Release Communications

This artifact should specify the different groups to which communications are directed, the method of communication (e.g., email, text or pager message, bulletin, newsletter, phone message, etc.). All communiques should be prepared in advance so that it is a matter of disseminating information when the release to production has been determined to be successful.

Also included in this artifact is a listing of the responsible parties who will execute the communications when a successful release has been declared (normally the Deployment Engineer), as well as the timing and dependencies of the communiques.

While there is no prescribed format for the release communications artifact, each communique should indicate the preferred delivery mechanisms (e.g., beeper notification, telephone calls, a posting to an internal release website, live or prerecorded presentations by senior management, etc.) and generally answer the following questions:

- Who are the parties (stakeholders) that are interested in knowing that a release to production has taken place?
- What specifically (features, functions, components) has been placed into production?

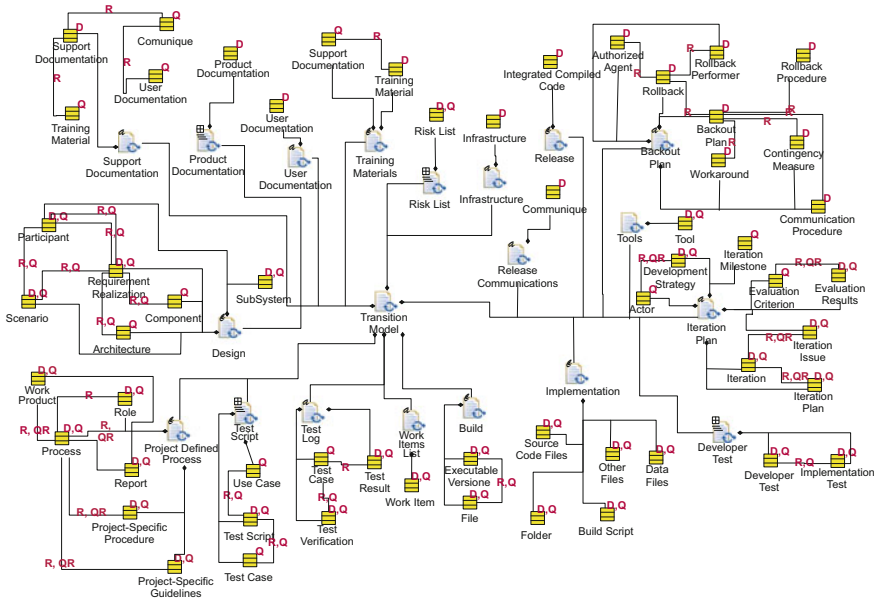

Fig. 55 The Transition phase documents structure

- Why is this release valuable to stakeholders and what business purpose does it serve?
- Where is the product available (on which platforms, geographical locations, business units, etc.)?
- How can the stakeholders access the system and under what circumstances?
- When was the product released (or when will it be released if the release date is in the future)?

Release Controls

Some common release controls are:

- A release or deployment plan must be documented and reviewed with the Deployment Manager (or the release organization). This plan must address how risks, issues, or code deviations are to be handled during the key timeframe leading up to the actual release.
- The components of each release package must be defined, integrated, and compatible with one another.
- The integrity of each release package must be verified and maintained.
- References to the configuration items (CIs) that the release package represents, if applicable.

Table 21 Transition phase—work product kinds

Name	Description	Work product kind
Backout Plan	A backout plan defines the criteria and procedures to be followed if a release into production needs to be rolled back.	Structured
Deployment Plan	See sections "Deployment Plan" and "WorkProduct Kinds."	Composite
Infrastructure	In reference to a release sprint, infrastructure refers to all the hardware, software, and network facilities necessary to support a deployed release. The purpose of this work product is to provide the underlying capabilities within which an application can be run as designed.	Free Text
Product Documentation	See sections "Product Documentation" and "WorkProduct Kinds."	
Release Communications	When a release is pushed to production, all the stakeholders of that product should be notified that the event has happened and what the release means to each of the stakeholders. Often, the output of this task does not need to be created from scratch; for products that plan multiple releases, just updating the communique details for each release might be enough. However, if any of the stakeholder groups change, or there is a significant difference in the product distribution, more significant content might need to be developed. In any case, communicating effectively to the end user community is important. A development team can develop high quality software, but if messaging to the stakeholders is conducted poorly or not at all, the end user experience might be degraded. By simply answering the questions "who, what, when, where, why, and how" in a format appropriate for each stakeholder group, a product release can become a more satisfying experience for all those involved.	Free Text
Release Controls	The purpose of this work product is to identify all the requirements to which a release package must conform to be considered "deployable."	Structured
Support Documentation	See sections "Support Documentation" and "WorkProduct Kinds."	
Training Materials	See sections "Training Materials" and "WorkProduct Kinds."	
User Documentation	See sections "User Documentation" and "WorkProduct Kinds."	

- The customer for which the application is being developed must approve the release, indicating that the user community (or a specific subset) is ready to receive and use the requisite capabilities of the release.
- Each release package must be capable of being backed out of production without negatively impacting the remaining production environment.
- The contents of each release package must be transferred to operations and support staff with sufficient documentation and knowledge transfer so that those organizations can effectively support the released capabilities in production.

3 Work Product Dependencies

Figure 56 describes the dependencies among the different work products and work product slots (see Sect. 1). The list of work products fulfilling the different work product slots is reported in Table 22.

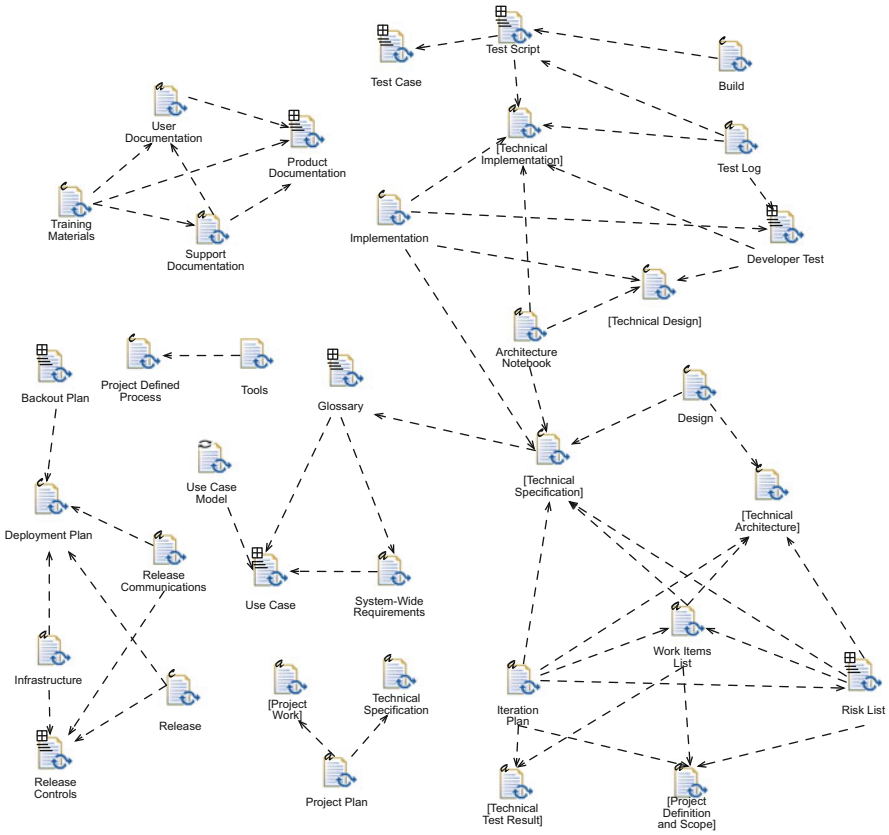

Fig. 56 The Work Product Dependency diagram

Table 22 OpenUp work product slots and the fulfilling work products

Work product slot	Fulfilling work product
[Project Definition and Scope]	Project Plan
[Project Work]	Iteration Plan, Work Items List
[Technical Architecture]	Architecture Notebook
[Technical Design]	Design
[Technical Implementation]	Build, Implementation
[Technical Specification]	Glossary, System-Wide Requirements, Use Case, Use-Case Model, Vision
[Technical Test Results]	Test Log

References

1. Eclipse Foundation: The OpenUP Website. Online at: http://epf.eclipse.org/wikis/openup/. Accessed on 15 Jan 2013
2. Eclipse Foundation: Work Product Slot. Online at: http://epf.eclipse.org/wikis/mam/core.mdev.common.base/guidances/concepts/work_product_slot_D5B44CE7.html. Accessed on 15 Jan 2013

3. Object Management Group (OMG): Unified Modeling Language (UML), V2.4.1, Infrastructure Specification. Doc. number: formal/2011-08-05 (2011)
4. Object Management Group (OMG): Unified Modeling Language (UML), V2.4.1, Superstructure Specification. Doc. number: formal/2011-08-06 (2011)
5. Kroll, P., Kruchten, P.: The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP. Addison-Wesley Professional, Boston (2003)
6. Software Process Engineering Metamodel. Version 2.0. Final Adopted Specification ptc/07 03-03. Object Management Group (OMG) (March 2007)
7. Seidita, V., Cossentino, M., Gaglio, S.: Using and extending the SPEM specifications to represent agent oriented methodologies. In: AOSE, pp. 46–59 (2008)