Appendices

The appendices draw together the resources for traceability that have been referred to throughout the book. Appendix A provides a copy of a traceability glossary that has been created by members of the traceability community. This glossary is included in an attempt to promote a greater consistency in the use of traceability terms and concepts, and we have attempted to remain faithful to this glossary where possible in the book. Appendices B and C provide the materials for the two case studies that have been used as exemplars in the various chapters. Appendix B provides a synopsis of the “iTrust Electronic Health Care System” and includes its associated resources. Appendix C does likewise for the “Mobile Phone Product Line Software System”. Appendix D provides an overview of “The Center of Excellence for Software Traceability”, and provides a link to its resources and membership opportunities. Appendix E lists the objectives of “TraceLab: A Tool for Supporting Traceability Research” and indicates how others can get involved.
Appendix A: Glossary of Traceability Terms (v1.0)

Orlena Gotel, Jane Cleland-Huang, Andrea Zisman, Jane Huffman Hayes, Alex Dekhtyar, Patrick Mäder, Alexander Egyed, Paul Grünbacher, Giuliano Antoniol, and Jonathan Maletic

**Answer set** – A known set of trace links derived prior to a tracing experiment, usually prepared by system experts.

**Artifact** – Something that is created or shaped by humans, either directly or indirectly via automation. In software and systems engineering contexts, the term refers to the products of the engineering process. See trace artifact.

**Artifact type** – See trace artifact type.

**Assisted traceability** – See semi-automated traceability.

**Assisted tracing** – See semi-automated tracing.

**Association** – An as yet unspecified connection between a pair of artifacts. Where augmented with semantics providing directionality, the association becomes traversable and is referred to as a trace link.

**Atomic trace** – A trace (noun sense) comprising a single source artifact, a single target artifact and a single trace link.

**Attribute** – A characteristic or property inherent in or ascribed to something. In software and systems engineering contexts, the term refers to the properties of artifacts and their trace links. See trace attribute.

**Automated traceability** – The potential for automated tracing.

**Automated tracing** – When traceability is established via automated techniques, methods and tools. Currently, it is the decision as to among which artifacts to create and maintain trace links that is automated.

**Backward traceability** – The potential for backward tracing.

**Backward tracing** – In software and systems engineering contexts, the term is commonly used when the tracing follows antecedent steps in a developmental path, which is not necessarily a chronological path, such as backward from code through design to requirements. Note that the trace links themselves could be used in either a primary or reverse trace link direction, dependent upon the specification of the participating traces.

This glossary is reproduced material from Center of Excellence for Software Traceability Technical Report #CoEST-2011-001, with permission. An up to date version of this glossary is maintained on the CoEST website (http://www.coest.org). Please direct any glossary additions or updates to this website. To promote consistency in the use of terms within the traceability community, preferred terms are denoted by * and U.S. English spellings are used throughout.
Bidirectional trace link – A term used to refer to the fact that a trace link can be used in both a primary trace link direction and a reverse trace link direction.

Bidirectional traceability – The potential for bidirectional tracing.

Bidirectional tracing – When tracing can be undertaken in both a forward and backward direction.

Body of knowledge for traceability – See Traceability Body of Knowledge (TBOK).

Candidate trace link – A potential, as yet unverified, trace link.

Center of Excellence for Software Traceability (CoEST) – A traceability community initiative. “Our goal is to bring together traceability researchers and experts in the field. We hope to encourage research collaborations, assemble a body of knowledge for traceability, and develop new technology to meet tracing needs.” (Hayes et al., 2007.) See: http://www.coest.org.

Chained trace – A trace (noun sense) comprising multiple atomic traces strung in sequence, such that a target artifact for one atomic trace becomes the source artifact for the next atomic trace.

Continuous traceability maintenance – The update of impacted trace links immediately following changes to traced artifacts.

Creating traceability – See traceability creation.

Element – A fundamental constituent of a composite entity. In a traceability context, the term refers to the fundamental constituents of a trace (noun sense). See trace element.

Establishing traceability – Enacting those parts of the traceability process associated with traceability creation and maintenance, and in accordance with the traceability strategy.

Forward traceability – The potential for forward tracing.

Forward tracing – In software and systems engineering contexts, the term is commonly used when the tracing follows subsequent steps in a developmental path, which is not necessarily a chronological path, such as forward from requirements through design to code. Note that the trace links themselves could be used in either a primary or reverse trace link direction, dependent upon the specification of the participating traces.

Golden standard requirements traceability matrix – See answer set.

Grand Challenge of Traceability – A fundamental problem with traceability that members of the international research and industrial communities agree deserves attention in order to achieve a revolutionary advance in traceability practice. It is a problem with no point solution; its solution involves first understanding and tackling a myriad of underlying challenges, and so will demand the effort of multiple research groups over an extended time period.

Horizontal traceability – The potential for horizontal tracing.

Horizontal tracing – In software and systems engineering contexts, the term is commonly used when tracing artifacts at the same level of abstraction, such as: (i) traces between all the requirements created by “Mary”, (ii) traces between requirements that are concerned with the performance of the system, or (iii)
traces between versions of a particular requirement at different moments in time. Horizontal tracing may employ both forward tracing and backward tracing.

Just in time tracing (JITT) – See reactive tracing.

Link – See trace link.

Link base – See link set.

Link semantics – The purpose or meaning of the trace link. The link semantics are generally specified in the trace link type, which is a broader term that may also capture other details regarding the nature of the trace link, such as how the trace link was created.

Link set – The totality of the trace links on a project.

Link type – See trace link type.

Maintaining traceability – See traceability maintenance.


Manual tracing – When traceability is established by the activities of a human tracer. This includes traceability creation and maintenance using the drag and drop methods that are commonly found in current requirements management tools.

Obsolete trace link – A pre-existing, and previously verified, trace link that is no longer valid.

On-demand traceability maintenance – A dedicated and overall update of the trace set (in whole or in part), generally in response to some explicit trigger and in preparation for an upcoming traceability use.

Post-requirements (specification) traceability – The potential for post-requirements (specification) tracing.

Post-requirements (specification) tracing – In software and systems engineering contexts, the term is commonly used to refer to those traces derived from or grounded in the requirements, and hence the traceability explicates the requirements’ deployment process. The tracing is, therefore, forward from requirements and back to requirements. Post-requirements (specification) tracing may employ forward tracing, backward tracing, horizontal tracing and vertical tracing.

Pre-requirements (specification) traceability – The potential for pre-requirements (specification) tracing.

Pre-requirements (specification) tracing – In software and systems engineering contexts, the term is commonly used to refer to those traces that show the derivation of the requirements from their original sources, and hence the traceability explicates the requirements’ production process. The tracing is, therefore, forward to requirements and back from requirements. Pre-requirements (specification) tracing may employ forward tracing, backward tracing, horizontal tracing and vertical tracing.

Primary trace link direction – When a trace link is traversed from its specified source artifact to its specified target artifact, it is being used in the primary direction as specified. Where link semantics are provided, they provide for a way to “read” the traversal (e.g., A implements B).
**Proactive tracing** – Initiating *trace capture* without explicit response to a stimulus to do so (i.e., *traces* are created in the background). Compare with *reactive tracing*.

**Prospective tracing** – See *trace capture*.

**Reactive tracing** – Responding to a stimulus to initiate *trace capture* (i.e., *traces* are created on demand). Compare with *proactive tracing*.

**Ready-to-use traceability** – Where previously established *trace links* are maintained as a project evolves, generally in compliance with a *traceability information model (TIM)*, so that the *traceability* on a project is always ready to be used according to the intentions for a project. This may combine *continuous* and *on-demand traceability maintenance* as appropriate.

**Reference set** – See *answer set*.

**Requirements management** – The activity concerned with the effective control of information related to stakeholder, system and software requirements and, in particular, the preservation of the integrity of that information for the life of the system and with respect to changes in the system and its environment. *Requirements management* depends upon *requirements traceability* as its enabling mechanism.

**Requirements management tools** – Tools that support *requirements management*.

**Requirements traceability** – “The ability to describe and follow the life of a requirement in both a forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases).” (Gotel and Finkelstein, 1994.)

**Requirements traceability matrix (RTM)** – See *traceability matrix*.

**Retrospective tracing** – See *trace recovery*.

**Reverse trace link direction** – When a *trace link* is traversed from its specified *target artifact* to its specified *source artifact*, it is being used in the reverse direction to its specification. The *link semantics* may no longer be valid, so a change from active to passive voice (or vice-versa) is generally required (e.g., if A replaces B then B is replaced by A).

**Semi-automated traceability** – The potential for *semi-automated tracing*.

**Semi-automated tracing** – When *traceability* is established via a combination of automated techniques, methods, tools and human activities. For example, automated techniques may suggest *candidate trace links* or *suspect trace links* and then the human *tracer* may be prompted to verify them.

**Software traceability** – See *requirements traceability*, extending the definition to encompass and interrelate any uniquely identifiable software engineering *artifact* to any other.

**Source artifact** – The *artifact* from which a *trace* originates.

**Stakeholder requirements for traceability** – *Stakeholder requirements for traceability* comprise two parts: (i) why end users (i.e., people, organizations, etc.) need *traceability*; and (ii) what *tracers* need in order to establish and use this *traceability*. The latter form part of the *system requirements for traceability*. 
Suspect trace link – A pre-existing, and previously verified, trace link that may no longer be valid.

System requirements for traceability – What the traceability solution needs to do to fulfill the stakeholder requirements for traceability. Note that the agent (human or automated) that establishes the traceability is part of the traceability solution.

Systems traceability – See requirements traceability, extending the definition to encompass and interrelate any uniquely identifiable systems engineering artifact to a broad range of systems-level components, such as people, processes and hardware models.

Target artifact∗ – The artifact at the destination of a trace.

Trace (Noun) – A specified triplet of elements comprising: a source artifact, a target artifact and a trace link associating the two artifacts. Where more than two artifacts are associated by a trace link, such as the aggregation of two artifacts linked to a third artifact, the aggregated artifacts are treated as a single trace artifact. The term applies, more generally, to both traces that are atomic in nature (i.e., singular) or chained in some way (i.e., plural).

Trace (Verb) – The act of following a trace link from a source artifact to a target artifact (primary trace link direction) or vice-versa (reverse trace link direction). See tracing.

Trace acquisition – See trace creation.

Trace artifact∗ – A traceable unit of data (e.g., a single requirement, a cluster of requirements, a UML class, a UML class operation, a Java class or even a person). A trace artifact is one of the trace elements and is qualified as either a source artifact or as a target artifact when it participates in a trace. The size of the traceable unit of data defines the granularity of the related trace.

Trace artifact type∗ – A label that characterizes those trace artifacts that have the same or a similar structure (syntax) and/or purpose (semantics). For example, requirements, design and test cases may be distinct artifact types.

Trace asset – See trace element.

Trace attribute∗ – Additional information (i.e., meta-data) that characterizes properties of the trace or of its individual trace elements, such as a date and time stamp of the trace’s creation or the trace link type.

Trace capture∗ – A particular approach to trace creation that implies the creation of trace links concurrently with the creation of the artifacts that they associate. These trace links may be created automatically or semi-automatically using tools.

Trace creation∗ – The activity of creating a single trace, associating two artifacts via a trace link. The trace link may be created manually, automatically using tools or semi-automatically using some combination of tool and manual input. The terms of trace capture, trace recovery and trace retrieval lend connotations as to when a trace link is created, along with the technique used to create the trace link in the case of trace retrieval.

Trace data – See trace element.

Trace element∗ – Used to refer to either one of the triplets comprising a trace: a source artifact, a target artifact or a trace link.
Trace generation – A particular approach to trace creation that implies that the trace links are created automatically or semi-automatically using tools.

Trace granularity – The level of detail at which a trace is recorded and performed. The granularity of a trace is defined by the granularity of the source artifact and the target artifact.

Trace life cycle – A conceptual model that describes the series of activities involved in the life of a single trace, from initial conception, through creation, maintenance and use, through to eventual retirement. This is the traceability process from the perspective of a single trace flowing through the traceability process.

Trace link* – A specified association between a pair of artifacts, one comprising the source artifact and one comprising the target artifact. The trace link is one of the trace elements. It may or may not be annotated to include information such as the link type and other semantic attributes. This definition of trace link implies that the link has a primary trace link direction for tracing. In practice, every trace link can be traversed in two directions (i.e., if A tests B then B is tested by A), so the link also has a reverse trace link direction for tracing. The trace link is effectively bidirectional. Where no concept of directionality is given or implied, it is referred to solely as an association.

Trace link type* – A label that characterizes those trace links that have the same or similar structure (syntax) and/or purpose (semantics). For example, “implements”, “tests”, “refines” and “replaces” may be distinct trace link types.

Trace maintenance – Those activities associated with updating a single pre-existing trace as changes are made to the traced artifacts and the traceability evolves, creating new traces where needed to keep the traceability relevant and up to date.

Trace precision – A commonly used metric in automated tracing that applies to represent the fraction of retrieved trace links that are relevant. It is computed as: \( \text{Precision} = \frac{\text{Relevant Links} \cap \text{Retrieved Links}}{\text{Retrieved Links}} \).

Trace quality – A measurable property of a single trace at a particular point in time on a project, such as a confidence score depicting its correctness.

Trace query – A term often used in the process of generating or vetting trace links, where one high level element is regarded as the trace query for searching into an artifact collection to find trace links (as distinguished from traceability-related queries).

Trace recall – A commonly used metric in automated tracing that applies to represent the fraction of relevant trace links that are retrieved. It is computed as: \( \text{Recall} = \frac{\text{Relevant Links} \cap \text{Retrieved Links}}{\text{Relevant Links}} \).

Trace record – Persistent information that registers the triplet of trace elements constituting a trace and is subject to version control. The trace record can also refer to the entire trace set.

Trace recovery* – A particular approach to trace creation that implies the creation of trace links after the artifacts that they associate have been generated and manipulated. These trace links may be created automatically or
semi-automatically using tools. The term can be construed to infer that the trace link previously existed but now is lost.

**Trace relation** – All the trace links created between two sets of specified trace artifact types. The trace relation is the instantiation of the trace relationship and hence is a collection of traces. For example, the trace relation would be the actual trace links that associate the instances of requirements artifacts with the instances of test case artifacts on a project. The trace relation is commonly recorded within a traceability matrix.

**Trace relationship** – An abstract definition of a permissible trace relation on a project (i.e., source artifact type, target artifact type and trace link types), as typically expressed within a traceability information model (TIM). Note that the trace links of the instances of the two artifact types may not necessarily have the same trace link type.

**Trace retrieval** – A particular approach to trace creation where information retrieval methods are used to dynamically create a trace link. This approach can be used for both trace capture and trace recovery.

**Trace set** – The totality of the traces on a project.

**Trace sink artifact** – See target artifact.

**Trace source artifact** – See source artifact.

**Trace target artifact** – See target artifact.

**Trace use** – Those activities associated with putting a single trace to use to support various software and systems engineering activities and tasks.

**Traceability** – The potential for traces to be established and used. Traceability (i.e., trace “ability”) is thereby an attribute of an artifact or of a collection of artifacts. Where there is traceability, tracing can be undertaken and the specified artifacts should be traceable.

**Traceability analyses** – The analyses that can be undertaken following traceability-related queries.

**Traceability benchmark** – A standard measure or test against which approaches to various aspects of the traceability process can be evaluated and compared.

**Traceability benchmark data** – Datasets that contain two or more artifact types and validated traceability matrices, the latter serving as answer sets (i.e., reference sets), for evaluating experimental results.

**Traceability Body of Knowledge (TBOK)*** – A proposed resource for the traceability community, containing traceability benchmarks, good traceability practices, traceability experience reports, etc.

**Traceability challenge** – A significant problem with traceability that members of the international research and industrial communities agree deserves attention in order to achieve advances in traceability practice.

**Traceability community** – Those people who are establishing and using traceability in practice, or have done so in the past or intend to do so in the future. Also, those people who are active in traceability research or in one of its many interrelated areas.

**Traceability configuration management** – The process of identifying, defining, recording and reporting on traces as configuration items, also controlling both
the release of *traces* for *traceability use* and the changes that occur during *traceability maintenance*. *Traceability configuration management* depends upon *traceability version control*.

**Traceability creation** – The general activity of associating two (or more) *artifacts*, by providing *trace links* between them, for *tracing* purposes. Note that this could be done manually, automatically or semi-automatically, and additional annotations can be provided as desired to characterize *attributes* of the *traces*.

**Traceability decay** – The gradual disintegration and break down of the *traceability* on a project. This tends to result following ongoing *traceability evolution*.

**Traceability-enabled activities and tasks** – Those software and systems engineering activities and tasks that *traceability* supports, such as verification and validation, impact analysis and change management.

**Traceability-enabled tasks and activities** – See *traceability-enabled activities and tasks*.

**Traceability end use** – See *traceability use*.

**Traceability end user** – The human or system engaged in *traceability use*.

**Traceability entropy** – The inevitable and steady deterioration of *traceability* as a result of *traceability decay*.

**Traceability evolution** – The gradual change of the *traceability* on a project. It generally refers to the tendency for pre-existing *traces* to become outdated and/or obsolete over time as changes are made to the *traced artifacts*, unless the *traceability* is maintained sufficiently. Ongoing deterioration of the *traceability* may lead to *traceability decay*.

**Traceability graph** – A representation of the *trace set*, with *trace artifacts* depicted as nodes and *trace links* depicted as edges.

**Traceability history** – A record of the *traceability evolution* and the associated *traceability maintenance* that has taken place on a project.

**Traceability information** – Any *traceability*-related data, such as *traceability information models*, *trace artifacts*, *trace links* and other *traceability work products*.

**Traceability information model (TIM)** – A graph defining the permissible *trace artifact types*, the permissible *trace link types* and the permissible *trace relationships* on a project, in order to address the anticipated *traceability-related queries* and *traceability-enabled activities and tasks*. The *TIM* is an abstract expression of the intended *traceability* for a project. The *TIM* may also capture additional information such as: the cardinality of the *trace artifacts* associated through a *trace link*, the *primary trace link direction*, the purpose of the *trace link* (i.e., the *link semantics*), the location of the *trace artifacts*, the *tracer* responsible for creating and maintaining the *trace link*, etc. (See Mäder et al. (2009) for more detail.)

**Traceability intent** – See *traceability information model (TIM)*.

**Traceability life cycle** – A conceptual model that describes the series of activities associated with a full end-to-end *traceability process*. 
**Traceability link** – A term often used in place of *trace link*. Arguably, while *traceability link* captures the enabling role of the *link* for *traceability* purposes, *trace link* emphasizes the fact that the *link* is a primary element of a *trace*.

**Traceability link document** – A document depicting *traces*, showing which pairs of *trace artifacts* are associated via *trace links*.

**Traceability maintenance** – Those activities associated with updating pre-existing *traces* as changes are made to the *traced artifacts* and the *traceability* evolves, *creating* new *traces* where needed to keep the *traceability* relevant and up to date.

**Traceability management** – Those activities associated with providing the control necessary to keep the *stakeholder* and *system requirements for traceability* and the *traceability solution* up to date during the life of a project. *Traceability management* is a fundamental part of *traceability strategy*.

**Traceability matrix** – A matrix recording the *traces* comprising a *trace relation*, showing which pairs of *trace artifacts* are associated via *trace links*.

**Traceability meta-model** – Defined constructs and rules related to the *trace artifact types* and *trace link types* for building *traceability information models* (TIMs).

**Traceability method** – A prescription of how to perform a collection of *traceability practices*, integrating *traceability techniques* with guidance as to their application and sequencing.

**Traceability metric** – A measure for some property or aspect of the *traceability process*, either quantitative or qualitative, such as *trace recall* and *trace precision* for *trace recovery*.

**Traceability model** – See *traceability information model* (TIM).

**Traceability network** – A *traceability graph* in which the directionality of the *trace links* is expressed (i.e., the *artifacts* are depicted as ordered pairs) and where the *trace links* are potentially weighted in some manner.

**Traceability planning** – Those activities associated with determining the *stakeholder* and *system requirements for traceability* and designing a suitable *traceability solution*. *Traceability planning* is a fundamental part of *traceability strategy*.

**Traceability policy** – Agreed principles and guidelines for *establishing* and using *traceability* in practice.

**Traceability practices** – Those actions and activities associated with *planning*, *managing*, *creating*, *maintaining* and using *traceability*.

**Traceability process** – An instance of a *traceability process model* defining the particular series of activities to be employed to establish *traceability* and render it usable for a particular project, along with a description of the responsibilities and resourcing required to undertake them, as well as their inputs and outputs. The *traceability process* defines how to undertake *traceability strategy*, *traceability creation*, *traceability maintenance* and *traceability use*.

**Traceability process improvement** – The activity of defining, analyzing and improving upon an existing *traceability process*. 
Traceability process model – An abstract description of the series of activities that serve to establish traceability and render it usable, along with a description of the typical responsibilities and resourcing required to undertake them, as well as their inputs and outputs. Distinctive steps of the process comprise traceability strategy, traceability creation, traceability maintenance and traceability use.

Traceability product – See traceability work products.

Traceability quality – A measurable property of the overall traceability at a particular point in time on a project, such as a confidence score depicting its overall correctness, accuracy, precision, completeness, consistency, timeliness, usefulness, etc.

Traceability quality assessment – The activity of assessing the traceability quality on a project.

Traceability quality assurance – The activity of assuring that defined standards and processes for traceability are appropriate and applied on a project.

Traceability quality attribute – A measurable property of a single trace link or of a group of trace links, such as a confidence score depicting the likelihood that a recovered candidate trace link is correct or the usefulness of a particular trace link over time.

Traceability reference model – See traceability information model (TIM).

Traceability-related queries – Those questions that a software or systems engineer may pose to which traceability can help to retrieve answers, such as the percentage of the specified requirements that are traceable to test cases and the existence of any requirements that are not traced through to design artifacts.

Traceability scheme – See traceability information model (TIM).

Traceability solution* – The traceability information model (TIM) and traceability process, as defined, designed and implemented for a particular project situation, along with any associated traceability tooling. The traceability solution is determined as a core part of the traceability strategy.

Traceability stakeholders – Those roles (i.e., people or systems) that have something to gain or something to lose from either having or not having traceability on a project.

Traceability standard – Mandatory practices and other conventions employed and enforced to prescribe a disciplined and uniform approach to traceability, generally written down and formed by consensus.

Traceability strategy – Those decisions made in order to determine the stakeholder and system requirements for traceability and to design a suitable traceability solution, and for providing the control necessary to keep these requirements and solutions relevant and effective during the life of a project. Traceability strategy comprises traceability planning and traceability management activities.

Traceability system – See traceability solution.

Traceability technique – A prescription of how to perform a single traceability practice, such as traceability creation, along with a description of how to represent its traceability work products.

Traceability tool – Any instrument or device that serves to assist or automate any part of the traceability process.
**Traceability use** – Those activities associated with putting *traces* to use to support various software and systems engineering activities and tasks, such as verification and validation, impact analysis and change management.

**Traceability version control** – Tracking changes to a particular *trace* over time. Each time a *trace* is changed in some way, a new version of the *trace* is effectively generated. This provides for an audit trail, and for parallel development and rollback possibilities.

**Traceability work products** – Those *artifacts* produced as a result of *planning*, *managing*, *creating*, *maintaining* and *using* *traceability*, including the *trace set*.

**Traceable** – The potential for *artifacts* to be accessed and retrieved by following *trace links* (i.e., by undertaking *tracing*). *Traceable* (i.e., *trace* “able”) is thereby an *attribute* of an *artifact* or of a collection of *artifacts*.

**Traced** – The *artifacts* that have been accessed by *tracing*, and so by having followed *trace links*.

**TraceLab** – A visual experimental workbench for designing and executing *traceability* experiments, providing *traceability* researchers with access to algorithms, datasets, experimental frameworks and benchmarking tools. *TraceLab* is a major component of the *Tracy project*.

**Tracer** – The agent engaged in the activity of *tracing*, where the agent can be a human or supporting tool.

**Tracing** – The activity of either *establishing* or *using* *traces*.

**Tracing activity or task** – A discrete and identifiable unit of work associated with the broader activity of *tracing*; an atomic activity of the *traceability process*.

**Tracing benchmark** – A clearly defined *tracing task*, with associated data sets and metrics that have been agreed upon by the *traceability community*, and which is used to evaluate different *traceability techniques* and *methods* comparatively.

**Tracing contest** – A clearly defined *tracing task* that has been identified by the *traceability community* as a critical *traceability practice* that warrants *traceability benchmarking*.

**Tracing task or activity** – See *tracing activity or task*.

**Tracking** – In software and systems engineering contexts, the term commonly applies to the act or process of following requirements and depends upon requirements *traceability*.

**Tracy project** – A National Science Foundation funded project designed to instrument the *traceability* research community, and to develop tools for facilitating the transfer of technology to industry and government organizations (Cleland-Huang et al., 2011).

**True requirements traceability matrix** – See *answer set*.

**Using traceability** – Enacting those parts of the *traceability process* associated with *traceability use*.

**Value-based traceability** – An approach to *traceability* that actively seeks to create, manage and measure either the monetary worth or utility worth of *traceability* on a project.

**Vertical traceability** – The potential for *vertical tracing*.
**Vertical tracing** – In software and systems engineering contexts, the term is commonly used when *tracing artifacts* at differing levels of abstraction so as to accommodate life cycle-wide or end-to-end *traceability*, such as from requirements to code. *Vertical tracing* may employ both *forward tracing* and *backward tracing*.

**References**


Appendix B: iTrust Electronic Health Care System Case Study

Andrew Meneely, Ben Smith, and Laurie Williams

1 Introduction

Electronic health record (EHR) systems present a formidable “trustworthiness” challenge because people’s health records, which are transmitted and protected by these systems, are just as valuable to a myriad of attackers as they are to health care practitioners. Major initiatives in EHR adoption and increased sharing of health information raise significant challenges for protecting the privacy of patients’ health information.

The United States is pursuing the vision of the National Health Information Network (NHIN) in which the electronic health records of the American people are passed between sometimes-competing health care providers. The American Recovery and Reinvestment Act of 2009 (ARRA, 2009) provides $34 billion of incentives to health care providers to deploy a government-approved EHR. The ARRA will, by 2014, impose penalties on those who do not. As a result, the use of EHR systems is likely to proliferate in the US in the next four years.

Dr. Laurie Williams created iTrust in 2005 as a course project for undergraduates in North Carolina State University’s Software Engineering course. iTrust is intended as a patient-centric application for maintaining an EHR. An ideal health care system combines medical information from multiple sources to provide a summary or detail view of the history of a particular patient in a way that is useful to the health care practitioner.

iTrust is not intended to fulfill the requirements set forth to be approved by the government, nor is it intended for use by practitioners in the field of medicine. The primary goal for the project is to provide software engineering students with a project with real-world relevance and enough depth and psychological complexity as to mimic industrial systems that students may encounter while working in the software industry. Additionally, iTrust provides an educational testbed for understanding the importance of security and privacy requirements. iTrust is particularly focused with maintaining the privacy standards set forth in the HIPAA Security and Privacy Rules (2002).

The notion that a software developer’s role is often to maintain, test, and refine software rather than creating it “from scratch” is a unique learning objective for students at North Carolina State. For the past five years, each semester students in
the undergraduate software engineering course enhance the project deemed to be the best from the prior semester. Refactoring of iTrust by graduate students often occurs during the summer. As such, students must learn the code base of more than 10,000 lines of Java Server Page code to make required enhancements.

This chapter highlights the key pieces of iTrust’s project artifacts that pertain to traceability and describes the project in detail. The version of iTrust we are describing in this chapter is v10.0, which was released in the August 8th, 2010, and built from requirements specification v18. The source code for this project, as well as all the artifacts we describe in this chapter are available from iTrust’s homepage. The iTrust project consists of the following artifacts:

- Source code, including:
  - Production source code (Java, Java Server Pages)
  - Automated test code
- Testing documents, including:
  - Black box test plan
  - Acceptance test plan
  - Test data
- Requirements, including sections describing:
  - System Roles
  - Use cases
  - Non-functional requirements and constraints
  - Data field formats
  - Use case tracing from requirements to JSP pages
- Traceability Matrix

The rest of this chapter is organized as follows. Section 2 focuses on iTrust as a project and how the team develops and maintains it. Section 3 describes an overview of the iTrust functionality. Section 4 describes the architecture and organization of the iTrust system. Section 5 describes the traceability provided by the project’s maintainers, and Section 6 summarizes the chapter.

2 iTrust Project

iTrust is an active team project for undergraduate students in North Carolina State University’s Software Engineering course. Dr. Laurie Williams conceived the project in the Fall of 2005 and the project has been released to undergraduate and graduate students at North Carolina State for the following five years (10 semesters).

1 http://realsearchgroup.com/iTrust
Table 1  iTrust project size

<table>
<thead>
<tr>
<th>Component</th>
<th>Number of files</th>
<th>LOC#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production classes</td>
<td>226</td>
<td>14,570</td>
</tr>
<tr>
<td>Java server pages</td>
<td>135</td>
<td>12,942</td>
</tr>
<tr>
<td>Unit tests</td>
<td>244</td>
<td>11,936</td>
</tr>
<tr>
<td>HTTP tests</td>
<td>50</td>
<td>4,146</td>
</tr>
</tbody>
</table>

As a part of their homework assignments, students in the undergraduate Software Engineering course as well as the graduate Software Testing course are required to perform maintenance and feature additions to iTrust.

In between semesters, the project administrators (typically graduate students) perform a “housekeeping” procedure. The graduate students spend approximately one to two weeks on housekeeping, and the procedure entails one or more of the following:

- **Updating the automated test plan**, which consists of improving the coverage and accuracy of JUnit and system-level integration testing.
- **Fixing or clarifying the documentation of the iTrust code**, which consists of Javadoc that explains the functionality and use of each Java class.
- **Discussions on the future of the project**, including possible architectural design changes, new decisions on technologies to use for testing, and other high-level decisions that would be infeasible during a semester.
- **Minor features**, which often involve removing or adding functionality that students have complained about but not changed, or functionality that would be required to prepare the system for assignments in the upcoming semester.
- **Cosmetic changes**, primarily involving editing the style sheets and Java Server Pages to improve the user interface of the system.
- **Refactoring**, which has often been major, involving a complete redesign of the system, or sometimes minor, such as implementing and redesigning a component of the system to be more amenable to future changes and development.

Table 1 presents measurements on the source lines of code and number of Java classes or JSP files that make up the iTrust code base.

The iTrust requirements v18 contains 40 functional requirements, six non-functional requirements, and eight constraints. iTrust v11 was released for download from SourceForge on August 8th, 2010. Since students were the primary developers for iTrust, there has been no public feedback on the project, although the install base is rather large. Since this release date, iTrust v11 has been downloaded from SourceForge 394 times.

### 3 iTrust Functionality

We designed iTrust to be a patient-centric application for maintaining an electronic health record. An ideal health record combines medical information from multiple sources to provide a summary or detail view of the history of a particular patient in
a way that is useful to the health care practitioner. iTrust is particularly focused with maintaining the privacy standards set forth in the HIPAA Security and Privacy Rules (2002). In addition to maintaining the patient’s personal information and health history, iTrust maintains a comprehensive transaction log. The transaction log, which can be used for repudiation and to track the actual operational profile, contains 53 different high-level transaction types that include viewing patients’ information, sending reminders, and adding a prescription. The patient can view a list of which health care professionals have viewed his or her medical information upon login. Also, iTrust has a focus on providing health care providers with dynamically determined information regarding a patient’s chronic disease risk factors including diabetes and heart disease. Finally, iTrust allows a health care professional to view trend information about patients’ causes of death. Often iTrust requirements are obtained from the US Department of Health and Human Services (HHS) use cases (2006–2009)\(^2\); those that are obtained from HHS reference the use cases. The remaining requirements are developed in a creative process by the teaching staff, with the intent of covering the software engineering curriculum.

### 3.1 System Roles

iTrust contains eight roles in its role-based access control system. The role of a user determines their viewing and editing capabilities.

- **Patient**: When an American infant is born or a foreigner requests medical care, each is assigned a medical identification number and password. Then, this person’s electronic records are accessible via the iTrust Medical Records system.
- **Administrator**: The administrator assigns medical identification numbers and passwords to LHCPs. (Note: for simplicity of the project, an administrator is added by directly entering the administrator into the database by an administrator that has access to the database.)
- **Licensed Health Care Professional (LHCP)**: A licensed health care professional that is allowed by a particular patient to view all approved medical records. In general, a patient does not know this non-designated health care professional, such as an emergency room doctor, and the set of approved records may be smaller than that granted to a designated licensed health care professional.
- **Designated Licensed Health Care Professional (DLHCP)**: A licensed health care professional that is allowed by a particular patient to view all approved medical records. Any LHCP can be a DLHCP to some patients (with whom he/she has an established relationship) and an LHCP to others (whom he/she has never/rarely seen before).
- **Emergency Responder (ER)**: Police, Fire, Emergency Medical Technicians (EMTs), and other medically trained emergency responders who provide care while at, or in transport from, the site of an emergency (referred to as “on site care

\(^2\) [http://www.hhs.gov/healthit/usecases/](http://www.hhs.gov/healthit/usecases/)
providers” by Department of Health and Human Services Emergency Responder Electronic Health Record Use Case (2006–2009)).

- **Unlicensed Authorized Personnel (UAP):** A health care worker such as a medical secretary, laboratory technician, case manager, care coordinator, or other authorized clerical-type personnel. An unlicensed personnel can enter and edit demographic information, diagnosis, office visit notes and other medical information, and can view records.

- **Personal Representative:** A person legally authorized to make health care decisions on an individual’s behalf or to act for a deceased individual. When a person logs into iTrust, if he or she is a personal representative, they view their own records or those of the person/people they are representing. (For example, a mother is a personal health representative for her children and could choose herself and any one of her children upon logging into iTrust.)

- **Public Health Agent:** A person legally authorized view and respond to aggregated reports of adverse events.

- **Software Tester:** An information technology worker who tests the iTrust Medical Records system. Of particular interest to the software tester is the operational profile information which informs him/her of the frequency of use of the features of the system.

### 3.2 Patient-Centered Functionality

One of the unique characteristics of iTrust is its patient-centered functionality where patients can log into the system to view their own records and perform a variety of tasks.

The primary way of tracking care for a given patient is through office visits. An office visit represents a specific consultation with an LHCP on a specific date in a specific location. Various standardized health care codes are linked to office visits, including diagnoses, immunizations, procedures, prescriptions, and general demographics such as height and weight. The LHCP logs the information for a given office visit, and the patient can view the records for of his or her previous office visits. Patients can also take a satisfaction survey on the LHCP, which is aggregated for other patients in search for an LHCP.

In addition to office visit tracking, patients have access to several forms of auditability. iTrust takes data provenance very seriously, so all access and changes to patient records are permanently logged. Patients are presented with an activity feed upon logging in to iTrust, and can configure email alerts when their records have been accessed or changed.

Lastly, iTrust focuses on providing informative feedback to both patients and LHCPs. Patients are shown potential risk factors on their record, such as for diabetes or heart disease. High risk patients who have not had a recent office visit are also alerted. LHCPs can also request biosurveillance to detect potential epidemics. The epidemic detection feature uses statistical modeling to determine an abnormal number of diagnoses for a given location. Additionally, LHCPs can view cause-of-death trends for a given location.
The requirements document in iTrust is a use-case based specification as shown in Fig. 1.

The requirements specification breaks down into the following sections:

- System Roles (described in Section 3.1)
- Use cases
Appendix B: iTrust Electronic Health Care System Case Study

- Non-functional requirements and constraints
- Data field formats

Each use case represents a small piece of functionality that students implemented in a two week iteration. The project administrators wrote the use cases in terms of the roles to imply the access controls surrounding the feature. Each use case has a precondition describing what conditions need to be met prior to accessing the feature (e.g. authentication). The main flow of the use case provides a high-level overview of the feature from the perspective of what the user does. The main flow of the use case references different sub-flows of the use case that provide added detail on the different events of the feature (e.g. the flow of events for when a patient is deceased). Lastly, each use case contains an alternative flow that describes the behavior of the feature outside of typical functionality (e.g. when the user enters wrong data). The requirements document also contains a reference from each sub-flow to the web page implementing that functionality. For an example of a use case, see the “Traceability in iTrust” section.

After the use cases, the rest of the document comprises of non-functional requirements and constraints. The non-functional requirements describe limitations that all features must adhere to. For example, all features must adhere to HIPAA standards. The constraints section covers the development process, such as the programming language and coding standards. iTrust was written in Java 1.5, and was designed to work with Tomcat v5.5.27 and MySQL 5.0.

The data field formats section covers all of the inputs to the iTrust system and how the field can be validated. For example, the data fields section defines which characters are allowed in a patient’s name. Many data fields are defined according to common health care standards. iTrust uses the following standard medical codes:

- ICD9CM for diagnoses
- CPT for procedures
- NDC for drug prescriptions

The iTrust requirements document is stored in a wiki format online. Storing the document in a wiki allows the requirements to be edited in a central location by authorized project maintainers. Each revision of the requirements document is retained so that the entire history of the document is preserved. Using the “diff” feature of the wiki also provides students with the ability to view what has recently changed in the requirements document without having to find changes manually.

4 iTrust Architecture

The iTrust source code is designed around the Model-View-Controller design pattern (Gamma et al., 1994). The goal of this organization is to separate the logic associated with the user interface (i.e. the “view”) from the logic of the persistent
storage (i.e. the “model”), while organizing most of the complex business logic in one place (i.e. the “controller”). In iTrust, the view is implemented in JavaServer Pages (JSPs), the controller is implemented in Java, and the model is implemented in SQL and Java. An overview of the iTrust architecture can be found in Fig. 2.

### 4.1 Source Code Organization

**View/JSPs.** The primary purpose of the JSPs is to provide a web-based user interface. Each JSP contains Java code, HTML, and potentially some Javascript. Each JSP has a one-to-one mapping to an action class. The JSP instantiates the Action class.

*Controller.* The overall purpose of the controller in iTrust is to provide a bridge between the user experience and the persistent storage of the database. Most of the complex logic behind validating data, and processing database query results are implemented in the controller.

The primary classes in the controller are *action classes*. Representing specific functionality in iTrust, the purpose of an action class is to delegate responsibility to the appropriate classes. Action classes serve as thin mediators between the user interface and the database and business logic. The responsibilities of action classes include:
- Delegating any input validation to a Validator.
- Logging transactions for auditability
- Delegating any custom business logic, such as risk factor calculations
- Delegating database interaction
- Handling exceptions in a secure manner

In addition to action classes, the controller contains validators. The sole purpose of validators is to validate any input brought into the system. Since security is a high priority in iTrust, the validators operate on using both whitelist and blacklist techniques for checking input. Additionally, the validators are designed to aggregate all errors in input so that the user is given a full report of all the problems with the input.

Lastly, the controller contains several classes with custom business logic. The custom business logic classes are a set miscellaneous Java classes designed for specific use cases. For example, Use Case 14 (UC14) is a feature for determining if a patient is at risk for several risk factors. Many of the queries involved in UC14 are specific to certain risk factors (e.g. having a viral infections during childhood), so the UC14 requires its own business logic.

**Model.** The model involves all of the logic related to persistent storage in iTrust. Beans are placeholders for data related to an iTrust entity (e.g. Patient). Beans have minimal functionality other than storing data. Other supporting classes load beans from database result sets, validate beans based on input, or any other custom logic needed.

The relational database is the sole storage mechanism for iTrust. The database stores all persistent information, including patient records, immunizations, office visits, and transaction logs. The database schema is defined by a set of custom scripts found in the source code tree. The database for iTrust does not contain any foreign keys, as the students who use iTrust do not usually have a background in relational databases and would not be able to debug foreign key constraint violations.

To interact with the database, iTrust employs database access objects (DAOs). DAOs are Java objects that interact with the iTrust relational database. Action classes will typically use DAOs to store and query the database. DAOs provide a set of common queries required by the action classes so that database query logic is contained to the DAO layer. Every DAO assumes that the incoming data is valid and any exception is handled by the Action classes. Connections to the DAOs are handled by the DAOFactory, which is a singleton class that utilizes a database connection pool for better performance and reliability. By convention, each database entity maps to a single Database Access Object and a single Bean.

### 4.2 Testing Artifacts

iTrust contains both automated and manual testing artifacts. All testing artifacts are constantly maintained throughout the development process.

**Black Box Test Plan.** As a part of their assignments in the graduate and undergraduate software engineering courses, students are required to maintain and
develop manual, black box test cases for the functionality of iTrust. The black box test plan is intended to be executed by a software tester using a web browser with no background in the project or how it can be used. The black box test plan is intended to cover each use case and sub-flow, including the exceptional or alternative flow cases.

A subset of the black box test plan is the acceptance test plan. The acceptance test plan is a set of black box, manual test cases that can be executed with passing results by the iTrust customer. When a new use case is developed for a course assignment, the instructors of the software engineering course develop an acceptance test case that corresponds to the use case. The acceptance test plan acts as a tool for grading how well students performed the assignment as well as providing a clarification of certain details of the specification that may be lacking from the requirements specification. Students are then responsible for adding additional black box tests for each use case flow.

Automated Unit Tests. The goal of the unit tests is to test individual iTrust functionality at the Java class level. Students are expected to test both regular functionality and boundary cases for virtually every unit in the iTrust system. When students are assigned faults to fix, they are required to write an automated unit test to ensure that the fault remains fixed. As the iTrust code is being developed, students are required to maintain 80% line coverage of all Java classes. Between semesters, the automated unit test plan is improved and maintained such that 80% coverage is maintained on all relevant classes if the students had not done so. Students are encouraged, but not required to use a test-driven approach to writing unit tests. iTrust uses JUnit for our automated unit tests, and EclEmma for code coverage in an Eclipse environment.

iTrust also contains a number of supporting classes to aid the automated testing process. A test database is set up clean before each unit test on database functionality (i.e. DAO classes), and the test data is a standard data set across all student projects.

There are some packages and classes of the iTrust Java classes for which unit testing does not make sense or is not applicable. The following types of classes are excluded from the 80% coverage requirement:

- **The Server Package**, which contains Java classes that interface with the Apache Tomcat API to provide session time out functionality and other web-server specific features.
- **Test Utilities**, which provide developer-friendly methods for inserting the correct test data into the database.
- **Tag Classes**, which provide custom JSP tags for data fields such as the US state the patient lives in.

Automated HTTP Tests. The automated HTTP tests simulate a user using iTrust in an web browser. Using HTTPUnit, the automated HTTP tests execute on a fully-deployed iTrust system by crafting HTTP requests and checking the responses.

3 http://httpunit.sourceforge.net/
As opposed to the automated unit tests, the automated HTTP tests are intended for regression testing. Students are required to implement HTTP tests based on the acceptance test plan. Thus, each acceptance test case is represented by at least one HTTP test. Students also automate security penetration testing using HTTP tests.

5 Traceability in iTrust

The iTrust project administrators maintain multiple traceability matrices amongst the artifacts. The main three artifacts that are involved in tracing are:

- Black box test plan
- Requirements document
- System archetypes (e.g. JSPs, Actions, Validators, DAOs)

Figure 3 shows an overview of how the test plan, requirements, and system archetypes are traced to each other. The requirements document contains sub-sections for each use case that trace to the implementing JSP. Students can use this traceability analysis to find the place in the code that implements a given requirement for comprehending the code as well as improved testing. Additionally, the whole traceability matrix is available for students on the wiki for posterity.

To construct the tracing, a software engineering graduate student conducted a manual traceability analysis on iTrust. The procedure was as follows:

![Fig. 3 Traceability overview](image-url)
1. Examine the first (or next) use case sub-flow in the iTrust requirements document. Record the unique identifier of the use case. For example, UC1S3.

2. Manually perform the action described in the use case. Record the relative URL in the browser window along with the use case and sub-flow. For example /hcp-uap/addPatient.jsp. The observed URLs correspond to JSP files (e.g. addPatient.jsp) contained within the iTrust code base.

3. If the use case cannot be performed, or does not involve any JSPs, enter “No links” for the use case and sub-flow.

4. Inspect the JSP code for the recorded URL. If more than one JSP is involved in executing the described action, for instance when more than one URL is observed in the browser window while executing the action, record each JSP separately on its own line with a trace to the use case and sub-flow in question.

5. Record DAOs, Action classes and Validators separately with their own trace to the sub-flow and use case in question.

6. Return to Step 1.

For an extended example of this traceability analysis, consider iTrust Use Case 1 sub-flow 1, whose traceability results are presented in Table 2.

**UC1. Create and disable patients use case**

**Preconditions:**
The iTrust HCP has authenticated himself or herself in the iTrust Medical Records system.

**Main Flow:**
An HCP creates patients and disables patients. The create/disable patients and HCP transaction is logged.

**Sub-flows:**

- [S1] The HCP enters a patient as a new user of iTrust Medical Records system. Only the name and email are is provided. An email with The patient’s assigned MID and a secret key (the initial password) is personally provided to the user, with which the user can reset his/her password. The HCP can edit the patient with all initial values (except patient MID) defaulting to null and/or 0 as appropriate. Patient MID should be the number assigned when the patient is added to the system and cannot be edited. The HCP does not have the ability to enter/edit/view the patient’s security question/password.

- [S2] The HCP provides the MID of a patient for whom he/she wants to disable. The HCP provides a deceased date. An optional diagnosis code is entered as the cause of death.

iTrust has a separately maintained list of manual black box test cases that students and administrators maintain. The black box test plan contained traceability to the requirements specification before the traceability analysis described in this chapter was complete. Students created and developed black box tests for the project as a
Table 2  Traceability results for Use Case 1 Sub-flow 1

<table>
<thead>
<tr>
<th>Use Case Subflow</th>
<th>Source Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC1S1</td>
<td>/auth/hcp-uap/addPatient.jsp</td>
</tr>
<tr>
<td>UC1S1</td>
<td>AddPatientAction().addPatient()</td>
</tr>
<tr>
<td>UC1S1</td>
<td>PatientDAO.addEmptyPatient()</td>
</tr>
<tr>
<td>UC1S1</td>
<td>AuthDAO.addUser()</td>
</tr>
<tr>
<td>UC1S1</td>
<td>PatientDAO.editPatient()</td>
</tr>
<tr>
<td>UC1S1</td>
<td>TransactionDAO.logTransaction()</td>
</tr>
<tr>
<td>UC1S2</td>
<td>No link</td>
</tr>
</tbody>
</table>

part of their course requirements and included the use case and sub-flow their test case was based upon when creating the test.

This traceability analysis procedure was scoped for the purposes of this case study, and is limited in the following ways:

1. The traceability was conducted manually. We did not look at possible automated approaches since we conducted the analysis exclusively for this case study.
2. The matrix was not checked and confirmed by any other students. Another researcher or developer performing the analysis may arrive at different results.

From the 40 functional requirements in the iTrust requirements specification v18, we elicited 199 use case sub-flows that could potentially trace to portions of the code. These 199 sub-flows contained 609 separate links to 310 Java methods or JSP files. Of the 199 use case sub-flows, 38 did not trace to any code within the iTrust project.

Although we traced the full list of 40 functional requirements, we excluded the set of six non-functional requirements in v18 of the iTrust requirements specification. The functional requirements typically traced to one or two components of each layer of the iTrust architecture. The traceability of the non-functional requirements in iTrust was less straightforward, however. Some of the non-functional requirements trace to every member of certain archetypes in iTrust (e.g. form validation), and others have no direct target (e.g. enabling multiple simultaneous users to be logged in).

6 Summary

iTrust is a patient-centered electronic health record web application used as an educational project in graduate and undergraduate software engineering courses at North Carolina State University. The software development project contains a use case-based requirements document, a black box test plan, automated tests, and source code. The project administrators maintain a manual traceability matrix from the black box test plan to the requirements document, and from the requirements document to the source code. iTrust is an open source software project, and all of its artifacts are publicly-available online.
References

Appendix C: Mobile Phone Product Line Software System Case Study

Waraporn Jirapanthong and Andrea Zisman

1 Introduction

We present in the following a case study for a line of software systems with different mobile phones. The mobile phone product line case study has been developed based on study, analysis, and discussions of mobile phone domains and ideas in http://www.forum.nokia.com/main.html; www.omg.org/technology/documents/formal/xmi.htm. This case study has also been used to evaluate the work in Jirapanthong and Zisman (2009). The various types of documents composing the case study are presented in details in this Appendix. It is worth noting that, when necessary, other documents were created by authors of the chapters in the book, based on the described functionalities of the mobile phone product line case study, to extend the case study and accommodate the need to illustrate or evaluate the works described in those chapters.

The mobile phone product line case study was developed using a feature-based object-oriented methodology. More specifically, we have used an extension of the FORM (Kang et al., 1998) methodology to develop the mobile phone system. A feature-based approach supports domain analysis and design, while an object-oriented approach assists with the development of various product members. The documents in the case study include feature, subsystem, process, and module models representing product line information; and use cases, class, and sequence diagrams representing information about product members. Table 1 presents a summary of the documents used in the case study. As shown in the table, these documents represent information in different phases of product line engineering namely domain engineering.
analysis and domain design, and different levels of specialisations in product line engineering namely product line and product member levels.

The line of systems in the case study contains three product members (mobile phones), namely PM_1, PM_2, and PM_3, with common and variable characteristics. Product member PM_1 is supposed to be a trendy device and is targeted at young people; product member PM_2 is intended to offer an elegant design and is targeted at business people; while product member PM_3 is targeted at users who enjoy media applications including games and music. Table 2 presents a summary of the various functionalities of these three product members.

**Table 2** List of functionalities of the product members

<table>
<thead>
<tr>
<th>Functionality</th>
<th>PM_1</th>
<th>PM_2</th>
<th>PM_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1: Make and receive calls using GSM 900</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>F2: Make and receive calls using GSM 1800</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>F3: Make and receive calls using GSM 1900</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>F4: Hold and swap a call</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>F5: Receive and update voice mail</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>F6: Display and update time and date</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>F7: Set alarm and time</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>F8: Record, display, and manipulate call logs</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>F9: Play games</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>F10: Update calendar</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>F11: Add, delete, and update preferences</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>F12: Add, delete, and update contacts</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>F13: Include calculator</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>F14: Take photos using VGA camera</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>F15: Take photos using VGA camera with 2x digital zoom</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>F16: FM radio</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F17: Email system using SMTP, POP3, or IMPA4</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>F18: Hand-free speaker</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F19: Send and receive text messages</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>F20: Send and receive multimedia message</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>F21: Play Real One format tunes and video</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F22: Play and record MP3 format tunes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F23: Record and update video (clips)</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F24: Play 3GPP video format</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>F25: Play Real Video format</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F26: Access Internet using WAP 1.2.1</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>F27: Access Internet using WAP 2.0</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F28: Access Internet using WAP XHTML</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>F29: Connect via Bluetooth transfer data</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>F30: Connect via Infrared transfer data</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>F31: Connect via USB</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F32: Play MIDI formatted tunes</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>F33: Play AMR formatted tunes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F34: Play AAC formatted tunes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F35: Play MP3 formatted tunes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F36: Play WAV formatted tunes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Functionality | PM_1 | PM_2 | PM_3
---|---|---|---
F37: Play True Tones formatted tunes |  | X |  
F38: Compose and play MIDI formatted ring tones | X | X |  
F39: Record and update voice messages | X | X | X  
F40: Transfer data via SyncML | X |  |  
F41: Transfer data via SyncML and TCP/IP |  |  | X  
F42: Support CLDC Java technology | X | X | X  
F43: Support MIDP Java technology | X | X | X  
F44: Support wireless messaging API Java technology | X | X |  
F45: Support mobile media API Java technology | X | X |  

We assume that for each line of software system being developed, there is a single instance of feature and subsystem models, but there may have various instances of process and module models and various instances of documents in the product member level (i.e., use cases, class, and sequence diagrams). This assumption is not unrealistic since the product line level represents general characteristics of a group of product members being developed, while the product member level is concerned with the various products in the group. Moreover, for a certain product line,
it is possible to have different behaviour for the subsystems represented by different process and module models, and for a certain product member, it is possible to have various ways of using and interacting with the product represented by different use cases, and sequence diagrams.

Table 3 shows a summary of the types and number of documents provided in the case study, and the size of the various documents with respect to the number of their main elements. For the documents representing information of product members (use cases, class, and sequence diagrams), we present the number of these documents and the number of the main elements in these documents for each product member in the case study.

In the following, we give a description of the various types of documents in the case study. The rest of the appendix is structured as follows. In Section 2 we present the product line level documents. In Section 3 we present the product member level documents. In Section 4 we show the feature model in XML format.

2 Product Line Level Documents

2.1 Feature Model

A feature model describes common and variable aspects (features) of a line of applications in a domain. In the FORM methodology (Kang et al., 1998), a feature model is composed of two parts: (a) a graphical hierarchy of features, and (b) a textual specification. Figure 1 presents the graphical representation of the feature model for the mobile phone case study. An example of the textual specification template proposed by the FORM methodology for Text Messages is presented in Fig. 2. The other textual descriptions are shown in XML format in Section 3.

As shown in Fig. 1, a feature is represented by a name and can be (i) mandatory, when it must exist in the applications in the domain; (ii) optional, when it is not necessary to be present in the applications in the domain; or (iii) alternative, when it can be selected for an application from a set of features that are related to the same parent feature in the hierarchy. The features can be classified into four groups, namely: (a) application capabilities, signifying features that represent functional aspects of the applications (e.g. calling, connectivity, personal preference, and tool features); (b) operating environments, signifying features that represent attributes of the environment in which product members are used and operated (e.g. network, input and output methods, and operating system features); (c) domain technologies, signifying features that represent specific implementation and technological aspects of the applications in the domain (e.g. WAP and XHTML2 browser types; specific Java application support like mobile media and wireless messaging application programming interface; SMTP, POP3, and IMAP43 network protocol features); and (iv) implementation techniques, signifying features that represent more general implementation and technological aspects of the applications, but not specific for the domain (e.g. PGP and DES encryption methods; AMR, MIDI, and MP3 sound formats; and 3GPP and MPEG4 video format features).
Fig. 1 Mobile phone case study feature model
Fig. 2 Textual template for feature model

Features can also be associated by different types of relationships. Examples of these relationships are (i) composed_of, (ii) generalisation/specialization, and (iii) implemented_by relationship types.

As shown in Fig. 2, the textual specification represents (i) a name, (ii) a description, (iii) issues and decisions representing trade-offs, rationale, or justifications for including the feature in an application, (iv) a type such as application capabilities, operating environments, domain technologies, and implementation technologies, (v) commonality indicating if a feature is mandatory, optional, and alternative, (vi) relationship with other features such as composed-of, implemented-by, generalisation/specialization, (vii) composition rule representing mutual dependency and mutual exclusion relationships to indicate consistency and completeness of a feature, if any, and (viii) allocated-to-subsystem indicating the name of a subsystem that contains the feature, if any.

2.2 Subsystem Model

A subsystem model is used at the product line level to represent the main functional groups of a system (internal subsystems), subsystems outside the scope of the system (external subsystems), and how the various subsystems relate to each other in terms of data and control flows. Figure 3 presents the subsystem model of the mobile phone case study composed of five subsystems, as described below.

(a) Operating System. This subsystem provides facilities for performing basic tasks such as control of the interaction with all devices, software, and data; support of the interaction between internal applications (e.g. games, multimedia, and PC connective), recognition of internal hardware (e.g. screen, keypad, and Bluetooth) and different types of input data (e.g. air signal, keystroke, screen touch, voice); response to different types of output data (e.g. air signal, screen-display, voice).

(b) Messaging. This subsystem manages the exchange and manipulation of messages. It supports two services: short message service (SMS) for textual messages, and multimedia message service (MMS) for multimedia messages.
The subsystem interacts with short message service centers (SMSC) or multimedia message service centers (MMSC) to receive and forward messages.

(c) Mobile Internet. This subsystem manages the interaction between wireless networks and tools such as plug-in applications (e.g. for online games and for mobile browser) and extra hardware (e.g. mobile game desk and 3G PCMCIA data card) for supporting mobile internet applications. The subsystem supports some special functionalities e.g. editing and browsing mobile web pages by using WML and XHTML techniques. The subsystem is also able to activate 24-hour connectivity and support mobile functions e.g. playing online games, managing personal online data, entertainment, and servicing online banking.

(d) Network. This subsystem supports the communication between different network protocols and the maintenance of network coverage for the mobile-phone devices. It manages a network protocol for transferring data over a mobile phone network e.g. GSM, GPRS, HSCSD, CSD and EDGE. It supports different network protocol architectures such as TCP, IPv4, IPv6, MSCHAP v2, IPSec, TCP/IP plug-in framework, WAP stack, and Multiple PDP context.

(e) Calling and Applications. This subsystem provides telephony management (e.g. creating and responding to phone calls), supports fundamental functions (e.g. a multimode API), and enables interworking of house-in applications (e.g. electronic games, clock and radio). In particular, the subsystem enables integration
of applications and the creation of advanced data services based on global network standards including GSM (Phase 2), GPRS (r4, Class B), CDMA2000 (1x), EDGE (ECSD, EGPRS), and WCDMA (r4).

2.3 Process Model

The dynamic behaviour of each subsystem in a subsystem model is represented in a graphical diagram called process model. A process model is composed of various processes, messages representing communication between the processes, and shared data used by the processes (e.g., databases, reports, files). A process can be resident, when it belongs to the subsystem, or transient, when it does not belong to the subsystem, but exchanges messages with a resident process. Processes can also be single or multiple, depending on the necessary number of instances of a process to perform a task.

Table 4 presents a list of all the process models and their respective processes for the five subsystems in the case study shown in Fig. 3. For the Messaging Subsystem there are two process models, namely process model for SMS (Short Messaging Service) and process model for MMS (Multimedia Messaging Service). The diagrams of the process models in the case study are presented in Figs. 4, 5, 6, 7, 8 and 9.

2.4 Module Model

Each process in a process model is further refined in a module model. A module model represents a hierarchical structure of the various modules composing a process and their interactions. The modules are classified into four groups related to the different groups of features, namely: (i) service modules, which support the functionality of the system and correspond to application capability features; (ii) environment hiding modules, which represent the running environment of the system and correspond to the operating environment features; (iii) technique hiding modules, which represent the technology domain aspects of the system and correspond to the domain technologies features; and (iv) utility modules, which represent general purpose aspects of the system and correspond to implementation techniques features.

We provide two module models in the case study, namely (a) module model for SMS for process model for SMS (messaging subsystem), and (b) module model for mobile internet for process model for internet subsystem. Figure 10 shows the module model for SMS. Table 5 presents a list of all the modules in this model. The module model for SMS contains 18 modules which are classified as 3 service modules, 1 environment handling module, 10 technique hiding modules, and 4 utility modules. Figure 11 shows the module model for mobile internet. Table 6 presents a list of all the modules in this model. The module model for mobile internet contains 22 modules which are classified as 4 service modules, 2 environment handling modules, 13 technique hiding modules, and 3 utility modules.
<table>
<thead>
<tr>
<th>Process model</th>
<th>Process name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process model for operating subsystem</td>
<td>• Recall reception process</td>
</tr>
<tr>
<td></td>
<td>• Establish the connection process</td>
</tr>
<tr>
<td></td>
<td>• Control functions process</td>
</tr>
<tr>
<td></td>
<td>• Display and interact process</td>
</tr>
<tr>
<td></td>
<td>• Edit information process</td>
</tr>
<tr>
<td></td>
<td>• Messaging service control process</td>
</tr>
<tr>
<td></td>
<td>• Making call control process</td>
</tr>
<tr>
<td></td>
<td>• Maintain logging process</td>
</tr>
<tr>
<td></td>
<td>• Invoke added-in application process</td>
</tr>
<tr>
<td></td>
<td>• Detect added-in hardware/software process</td>
</tr>
<tr>
<td></td>
<td>• IRQ (Interrupt ReQuest) process</td>
</tr>
<tr>
<td></td>
<td>• IPC (Inter Process Communication) process</td>
</tr>
<tr>
<td>Process model for SMS (Messaging subsystem)</td>
<td>• Short messaging control process</td>
</tr>
<tr>
<td></td>
<td>• Check signal process</td>
</tr>
<tr>
<td></td>
<td>• Edit process</td>
</tr>
<tr>
<td></td>
<td>• Short Messaging Service (SMS) control process</td>
</tr>
<tr>
<td></td>
<td>• Short Messaging Service Center (SMSC) process</td>
</tr>
<tr>
<td></td>
<td>• Notification process</td>
</tr>
<tr>
<td></td>
<td>• Update remotely process</td>
</tr>
<tr>
<td>Process model for MMS (Messaging subsystem)</td>
<td>• Multimedia messaging control process</td>
</tr>
<tr>
<td></td>
<td>• Check signal process</td>
</tr>
<tr>
<td></td>
<td>• Edit process</td>
</tr>
<tr>
<td></td>
<td>• Multimedia Messaging Service (MMS) control process</td>
</tr>
<tr>
<td></td>
<td>• Multimedia Messaging Service Center (MMSC) process</td>
</tr>
<tr>
<td></td>
<td>• Notification process</td>
</tr>
<tr>
<td></td>
<td>• Update remotely process</td>
</tr>
<tr>
<td>Process model for mobile internet subsystem</td>
<td>• Trigger process</td>
</tr>
<tr>
<td></td>
<td>• Download software process</td>
</tr>
<tr>
<td></td>
<td>• Launch application process</td>
</tr>
<tr>
<td></td>
<td>• Restore data process</td>
</tr>
<tr>
<td></td>
<td>• Maintain reception process</td>
</tr>
<tr>
<td></td>
<td>• Control process</td>
</tr>
<tr>
<td>Process model for network subsystem</td>
<td>• Establish high-range signal process</td>
</tr>
<tr>
<td></td>
<td>• Check authentication process</td>
</tr>
<tr>
<td></td>
<td>• Valid equipment process</td>
</tr>
<tr>
<td></td>
<td>• Find signal process</td>
</tr>
<tr>
<td></td>
<td>• Forward signals process</td>
</tr>
<tr>
<td></td>
<td>• Register subscriber process (Roaming)</td>
</tr>
<tr>
<td></td>
<td>• Handoff process</td>
</tr>
<tr>
<td></td>
<td>• Establish low-range signal process</td>
</tr>
<tr>
<td>Process model for calling subsystem</td>
<td>• Calling control process</td>
</tr>
<tr>
<td></td>
<td>• Compose a call process</td>
</tr>
<tr>
<td></td>
<td>• Keep logging process</td>
</tr>
<tr>
<td></td>
<td>• Check a signal/ reception process</td>
</tr>
<tr>
<td></td>
<td>• Trigger a receiving call process</td>
</tr>
<tr>
<td></td>
<td>• Delivery a call process</td>
</tr>
<tr>
<td></td>
<td>• Forwarding a call to voice mail process</td>
</tr>
<tr>
<td></td>
<td>• Accepting a call process</td>
</tr>
</tbody>
</table>
Recall reception process
Establish the connection process
Control functions process
Display and interact process
Edit information process
Detect added-in hardware/software process
Profile contacts
Log file
MQ
Detect added-in hardware/software process
Invoke added-in application process
IRQ
IPC
IRQ data
Messages
Recall reception process
Establish the connection process
Control functions process
Display and interact process
Edit information process
Detect added-in hardware/software process
Profile contacts
Log file
MQ
Detect added-in hardware/software process
Invoke added-in application process
IRQ
IPC
IRQ data
Messages
Fig. 4 Process model for operating subsystem

Update remotely process
Edit process
SMS template
Short messaging control process
Check signal process
Notification process
Fig. 5 Process model for messaging subsystem (SMS)
Appendix C: Mobile Phone Product Line Software System Case Study

**Fig. 6** Process model for messaging subsystem (MMS)

**Fig. 7** Process model for mobile internet subsystem
Appendix C: Mobile Phone Product Line Software System Case Study

Fig. 8  Process model for network subsystem
Profile contacts

Process model for Calling Telephony subsystem

Forwarding a call to voice mail process

Accepting a call process

Trigger a receiving call process

Call process

Delivery a call process

Compose a call process

Check a signal/reception process

Profile contacts

Fig. 9 Process model for calling subsystem

Module model for Messaging

Phone system

Service

Messaging controller

Connecting

Data controller

Environment Hiding

Multi network

Technique Hiding

Input Interface

Output Interface

Input/Output Interface

Signalling controller

ID Interface controller

Edit controller

Utility

Text display

Web display

Timer

Data encryption

Fig. 10 Module model for short messaging service (SMS)
### Table 5  List of modules in short messaging service SMS module model

<table>
<thead>
<tr>
<th>Module</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messaging controller</td>
<td>Pre-coded</td>
<td>Controls messages</td>
</tr>
<tr>
<td>Connecting</td>
<td>Pre-coded</td>
<td>Establishes a network communication</td>
</tr>
<tr>
<td>Data controller</td>
<td>Pre-coded</td>
<td>Controls internal data of mobile-phone handset</td>
</tr>
<tr>
<td>Multi-network</td>
<td>Pre-coded</td>
<td>Responds to multi-networks</td>
</tr>
<tr>
<td>Signaling controller</td>
<td>Template</td>
<td>Provides algorithms for maintaining the mobile-phone reception and supporting different mobile-phone networks</td>
</tr>
<tr>
<td>IO Interface controller</td>
<td>Pre-coded</td>
<td>Provides software interfaces for input and output devices of a mobile-phone handset</td>
</tr>
<tr>
<td>Edit controller</td>
<td>Pre-coded</td>
<td>Manages editors</td>
</tr>
<tr>
<td>Output Interface</td>
<td>Skeleton</td>
<td>Manages output devices of a mobile-phone handset</td>
</tr>
<tr>
<td>Input/Output Interface</td>
<td>Skeleton</td>
<td>Manages input and output devices of a mobile-phone handset</td>
</tr>
<tr>
<td>Input Interface</td>
<td>Skeleton</td>
<td>Manages input devices of a mobile-phone handset</td>
</tr>
<tr>
<td>Display</td>
<td>Pre-coded</td>
<td>Displays data to output devices of a mobile-phone handset</td>
</tr>
<tr>
<td>Touch screen</td>
<td>Pre-coded</td>
<td>Manages touch screen of a mobile-phone handset</td>
</tr>
<tr>
<td>Keypad</td>
<td>Pre-coded</td>
<td>Manages a keypad of a mobile-phone handset</td>
</tr>
<tr>
<td>Joystick</td>
<td>Pre-coded</td>
<td>Manages a joystick of a mobile-phone handset</td>
</tr>
<tr>
<td>Textual display</td>
<td>Pre-coded</td>
<td>Manages a textual display of a mobile-phone handset to support displaying text</td>
</tr>
<tr>
<td>Web display</td>
<td>Pre-coded</td>
<td>Manages a graphical display of a mobile-phone handset to support displaying web pages</td>
</tr>
<tr>
<td>Timer</td>
<td>Pre-coded</td>
<td>Sets and displays the time</td>
</tr>
<tr>
<td>Data encryption</td>
<td>Pre-coded</td>
<td>Encrypts and decrypts data</td>
</tr>
</tbody>
</table>

![Module model for Mobile internet](image)

**Fig. 11** Module model for mobile internet process model
Table 6  List of modules in mobile internet process model

<table>
<thead>
<tr>
<th>Module</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application controller</td>
<td>Pre-coded</td>
<td>Controls a running (local) application</td>
</tr>
<tr>
<td>Connecting</td>
<td>Pre-coded</td>
<td>Establishes a network communication</td>
</tr>
<tr>
<td>Data controller</td>
<td>Pre-coded</td>
<td>Controls internal data of mobile-phone handset</td>
</tr>
<tr>
<td>Mobile-phone Internet</td>
<td>Pre-coded</td>
<td>Controls a running Internet application</td>
</tr>
<tr>
<td>Multi-network controller</td>
<td>Pre-coded</td>
<td>Responds to multi-networks</td>
</tr>
<tr>
<td>Multi-platform</td>
<td>Pre-coded</td>
<td>Responds to multi-platform applications</td>
</tr>
<tr>
<td>Signaling controller</td>
<td>Template</td>
<td>Provides algorithms for maintaining the mobile-phone reception</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and supporting different mobile-phone networks</td>
</tr>
<tr>
<td>IO Interface controller</td>
<td>Pre-coded</td>
<td>Provides software interfaces for input and output devices</td>
</tr>
<tr>
<td>WAP controller</td>
<td>Pre-coded</td>
<td>Controls WAP browsing</td>
</tr>
<tr>
<td>Emailing</td>
<td>Template</td>
<td>Provides algorithms for composing emails and supporting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>different emailing protocols</td>
</tr>
<tr>
<td>JavaTM support technique</td>
<td>Template</td>
<td>Manages Java-based plug-ins</td>
</tr>
<tr>
<td>Device Interface</td>
<td>Skeleton</td>
<td>Manages interfaces for different devices of mobile-phone</td>
</tr>
<tr>
<td></td>
<td></td>
<td>handsets e.g. game desk, PDA, computers</td>
</tr>
<tr>
<td>Output Interface</td>
<td>Skeleton</td>
<td>Manages output devices of a mobile-phone handset</td>
</tr>
<tr>
<td>Input/Output Interface</td>
<td>Skeleton</td>
<td>Manages input and output devices of a mobile-phone handset</td>
</tr>
<tr>
<td>Input Interface</td>
<td>Skeleton</td>
<td>Manages input devices of a mobile-phone handset</td>
</tr>
<tr>
<td>Display</td>
<td>Pre-coded</td>
<td>Displays data to output devices of a mobile-phone handset</td>
</tr>
<tr>
<td>Touch screen</td>
<td>Pre-coded</td>
<td>Manages a touch screen of a mobile-phone handset</td>
</tr>
<tr>
<td>Keypad</td>
<td>Pre-coded</td>
<td>Manages a keypad of a mobile-phone handset</td>
</tr>
<tr>
<td>Joystick</td>
<td>Pre-coded</td>
<td>Manages a joystick of a mobile-phone handset</td>
</tr>
<tr>
<td>Web display</td>
<td>Pre-coded</td>
<td>Manages a graphical display of a mobile-phone handset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to support displaying web pages</td>
</tr>
<tr>
<td>Timer</td>
<td>Pre-coded</td>
<td>Sets and displays the time</td>
</tr>
<tr>
<td>Data encryption</td>
<td>Pre-coded</td>
<td>Encrypts and decrypts data</td>
</tr>
</tbody>
</table>

3 Product Member Level Documents

The three product members in the case study (PM_1, PM_2, and PM_3) are designed and documented in terms of use cases and UML class and sequence diagrams. Use cases are used to represent the functional requirements of the products. One product member can have several use cases. The use cases are specified in natural language following a template that is a variant of the one proposed in Cockburn (1997). The design aspects of the product members are represented as class and sequence diagrams. Each product member has one class diagram, but can have several sequence diagrams. We present below the documents for this case study.
3.1 Product Member PM_1

Use Cases: The four use cases for product member PM_1 are: UC1: Making a call, UC2: Taking a photo, UC3: Sending emails, and UC4: Transferring data. These use cases are described below.

Use Case UC1: Making a call

Status: Common
Region: EU, Africa, Asia Pacific

CHARACTERISTIC INFORMATION
Description: The phone is able to make a call. The user can select a calling phone number from a list of phone numbers, which are restored in the data collection, or enter the number via the keypad. After the user confirms the call, the phone establishes the line connection to create the call. If properly done, the phone dials for a response from the receiver. Otherwise, the phone informs the user of a problem on the connection. In the case that the destination number is engaged, or it is not able to establish a signal, the phone responds with a voice message.

Level: Primary task
Preconditions: The user has selected the function for making a call from the main menu.
Postconditions: The user has finished a call. The phone is ready for next actions.
Primary Actor: The user
Secondary Actors: -

FLOW OF EVENTS
Trigger:
1. The system is ready to make a call.
2. The user selects a phone number from the list of contacts or enters a phone number via keypad.
3. The user confirms making a call.
4. The system establishes the line connection.
5. If the connection is properly set, the phone dials the number to the destination. Otherwise, the phone informs the user about existing problems.
6. If the destination number is engaged or not able to be reached, the phone informs the user.
7. The user confirms by hanging up the call.
8. The phone disconnects.
9. The phone shows the attempt to make a call to the user.
10. The phone keeps a log file of calls made in the data storage.
Use Case UC2: Taking a photo

Status: Common
Region: EU, Africa, Asia Pacific

CHARACTERISTIC INFORMATION
Description: The phone is integrated with a digital camera. It enables a user to take and restore pictures from the phone. The photo file is in JPG format. The photo can be taken as one of three optional types: general, night, and portrait. Each of these types are of different size. The pictures stored in the phone can be viewed and deleted afterwards.
Level: Primary task
Preconditions: The user has selected a function for taking a photo from the main menu.
Postconditions: The phone has taken a photo and kept it as a JPG-formatted file in its temporary memory storage in order to be restored in the data collection later on. The phone is ready for capturing future shots.
Primary Actor: The user
Secondary Actors: -

FLOW OF EVENTS
Trigger:
1. The system shows a list of photo types, i.e. general, night, and portrait.
2. The user selects one of the photo types.
3. The system shows the scenario on the screen.
4. The user clicks the button on the phone to capture a snapshot.
5. The system displays the shot that has just been taken.
6. The system pops up a request for restoring the snapshot as a photo in the phone.
7. If the user wants to keep the snapshot, the system restores the photo as a JPG file in the data collection.
8. The system shows the scenario on the screen for other snapshots.
EXCEPTIONAL EVENTS
None identified at present.

RELATED INFORMATION
Superordinate Use Case: None
Subordinate Use Cases: None

Use Case UC3: Sending emails

Status: Common
Region: EU, Africa, Asia Pacific

CHARACTERISTIC INFORMATION
Description: The phone is able to send emails with attachment using different network protocols such as SMTP, POP3, IMAP4. The user can specify the receiver’s email addresses by selecting them from a contact list, which are stored in the data collection of the phone, or entered via keypad. The phone can send emails to multiple receivers. The user can attach different file types to the emails including images and photos. The phone keeps a log file of emails sent in the data storage. The user can view and delete the log file later on.
Level: Primary task
Preconditions: The user has selected a function for sending emails from the main menu.
Postconditions: The phone sends the email to specified receivers and shows a confirmation to the user.
Primary Actor: The user Secondary Actors: -

FLOW OF EVENTS
Trigger:
1. The system shows an editor composed of a text box for specifying the email addresses of the receivers and a blank note for writing a message.
2. The user inserts a receiver’s email address by selecting it from a contact list stored in the data collection of the phone, or entering it via keypad. Note that the user can send the email to multiple receivers by separating the email addresses with ‘;’.
3. The user can type the message.
4. The user may attach files or notes (.txt), photos (.jpg), and images (.jpg) to the email that are available in the phone. (Note that the event of 2, 3, and 4 are not sequential processes.)
5. The user confirms by sending the email.
6. The phone establishes the connection for sending emails.
7. If the connection is properly set, the phone sends the email via network protocols. Otherwise, the phone informs the user about any problems.
8. After the email is sent, the phone disconnects.
9. The phone shows that an email has been sent and keeps this information in a log file of the phone.

--------------------
EXCEPTIONAL EVENTS
None identified at present.
--------------------

RELATED INFORMATION
Superordinate Use Case: None     Subordinate Use Case: None.

Use Case UC4: Transferring data

Status: Common
Region: EU, Africa, Asia Pacific

-------------------------------
CHARACTERISTIC INFORMATION
Description: The phone is able to transfer data that are stored in the data collection of the phone via communication ports such as Bluetooth and infrared to another device attached with the same communication port. For example, the user can transfer photos taken by an integrated camera to a computer being attached with a Bluetooth device. Data can be any sort of files or notes (.txt), photos (.jpg), and images (.jpg). The phone can transfer one data item at a time.
Level: Primary task
Preconditions: The user has already selected a function for transferring data.
Postconditions: The system displays the status of data transferred.
Primary Actor: The user
Secondary Actors: -

-------------------------------
FLOW OF EVENTS
Trigger:
1. The user selects a data item to be transferred from the data collection.
2. The system shows a list of communication ports for transferring the data item. For PM_1, there are two ports Bluetooth and infrared.
3. The user selects a communication port.
4. The system searches a destination port from a device that is closest to the phone for transferring the data item.
5. If the destination port is found, the system establishes the communication channel between the phone and the device via the communication port. Otherwise, the phone notifies the user about the problem.
6. The system transfers the data item.
7. While transferring, the phone displays the status of transferring the data item on the screen.
8. After completed, the phone disconnects.
9. The phone shows the status of data item transferred.

---------------
EXCEPTIONAL EVENTS
None identified at present.
---------------
RELATED INFORMATION
Superordinate Use Case: None
Subordinate Use Cases: None
Class Diagram

Fig. 12 Class diagram for product member PM_1
Sequence Diagrams

Fig. 13 Making a call sequence diagram
Fig. 14  Taking a photo sequence diagram
Fig. 15 Sending message sequence diagram
Fig. 16 Transferring data sequence diagram
3.2 Product Member PM_2

Use Cases: The four use cases for PM_2 are: UC5: Making a call, UC6: Taking a photo, UC7: Sending emails, and UC8: Transferring data. Use cases UC5, UC6, and UC7 are the same as use cases UC1, UC2, and UC3 for product member PM_1, respectively. Use case UC8 is slightly different than its respective use case UC4. It is described below.

<table>
<thead>
<tr>
<th>Use Case UC8: Transferring data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status:</strong> Common</td>
</tr>
<tr>
<td><strong>Region:</strong> EU, Africa, Asia Pacific</td>
</tr>
</tbody>
</table>

CHARACTERISTIC INFORMATION

Description: The phone is able to transfer data that are stored in the data collection of the phone via communication channels such as Bluetooth, infrared, or TCP/IP network to another device attached with the same communication channels. For example, the user can transfer photos taken by an integrated camera to a computer being attached with a Bluetooth device. The user can also transfer a text file via local network working on TCP/IP protocol. Data can be any sort of files or notes (.txt), photos (.jpg), and images (.jpg). The phone can transfer one data item at each time.

Level: Primary task

Preconditions: The user has already selected a function for transferring data.

Postconditions: The system displays the status of transferred.

Primary Actor: The user

Secondary Actors: -

FLOW OF EVENTS

Trigger:

1. The user selects a data item to be transferred from the data collection.
2. The system shows a list of communication channels for transferring the data item. For PM2, there are not only communication ports Bluetooth and infrared, but the phone is also able to transfer data via the local network using TCP/IP protocol.
3. The user selects a communication channel.
4. The system searches a destination channel from a device that is closest to the phone for transferring the data item.
5. If the channel is found, the system establishes the communication channel between the phone and the device via the communication port. Otherwise, the phone notifies the user about the problem.
6. The system transfers the data item.
7. While transferring, the phone displays the status of transferring the data item on the screen.
8. After completed, the phone disconnects.
9. The phone shows the status of data item transferred.

-------------------
EXCEPTIONAL EVENTS
None identified at present.
-------------------
RELATED INFORMATION
Superordinate Use Case: None
Subordinate Use Cases: None
Class Diagram

Fig. 17 Class diagram for product member PM_2
**Sequence Diagram:** The sequence diagrams for product member PM_2 are the same as the sequence diagrams for product member PM_1. An exception is found in the transferring data sequence diagram, which is extended to support data to be transferred using TCP/IP protocol.

### 3.3 Product Member PM_3

**Use Cases:** The three use cases for product member PM_3 are: UC9: Making a call, UC10: Sending emails, and UC11: Transferring data. Use cases UC9, UC10, and UC11 are the same as use cases UC1, UC3, and UC4, respectively for product member PM_1.

**Sequence Diagram:** The sequence diagrams for product member PM_3 are the same as the sequence diagrams for PM_1, for making a call, sending emails, and transferring data.
Class Diagram

Fig. 18 Class diagram for product member PM_3
4 Feature Model in XML Format

<Feature_Model
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Feature>
    <Feature_name>Bluetooth</Feature_name>
    <Description>Bluetooth enables cost-free wireless connections between electronic devices within various maximum ranges according to the models of phones. A Bluetooth connection can be used to send images, texts, business cards, calendar notes, or to connect wirelessly to Bluetooth enabled devices such as computers. Since Bluetooth devices communicate using radio waves the phone and the other Bluetooth device do not need to be in direct line-of-sight. The two devices only need to be within various maximum of distance depending to the models of the phones. The connection can be subject to interference from obstructions such as walls or from other electronic devices. Moreover using Bluetooth consumes the battery and the phone’s operating time will be reduced.</Description>
    <Issue_and_decision />
    <Type />
    <Existential>Optional</Existential>
  </Feature>
  <Feature>
    <Feature_name>Digital Camera</Feature_name>
    <Description>With the camera customers can take photo or events while on the move. The photos are automatically saved in the Images application where the customers can rename them and organise them in folders. The customers can also send data to people in a multimedia message as an e-mail attachment or via a Bluetooth or infrared connection. The camera produces JPEG photos.</Description>
    <Issue_and_decision />
    <Type />
    <Existential>Optional</Existential>
    <Relationship type="Composed-Of">
      <Rel_feature>Taking Photos</Rel_feature>
    </Relationship>
  </Feature>
  <Feature>
    <Feature_name>Edit Text Message</Feature_name>
    <Description>Customers can compose and send the message through network.</Description>
    <Issue_and_decision />
    <Type />
    <Existential>Mandatory</Existential>
    <Relationship />
    <Allocated_to_Subsystem>Messaging</Allocated_to_Subsystem>
  </Feature>
</Feature_Model>
- <Feature>
  <Feature_name>Taking Photos</Feature_name>
  <Description>With an integrated digital camera the phone can take photos. The Camera application is opened and customers can see the view to be captured. The screen shows the viewfinder and the cropping lines which shows the photo area to be captured. The customers can also see the image counter which shows how many images depending on the selected image quality fit in the memory of the phone. The lens range is various according to the models of phones. If the distance to the subject of taking a photo is closer than the minimum distance of the lens range it may affect the sharpness of images. The photos are saved automatically in the Images application.</Description>
  <Issue_and_decision />
  <Type>Capability</Type>
  <Existential>Mandatory</Existential>
  <Relationship />
</Feature>
- <Feature>
  <Feature_name>Connectivity</Feature_name>
  <Description>The phone must be switched on to use the functions in the Connectivity folder. Do not switch the phone on when wireless phone use is prohibited or when it may cause interference or danger.</Description>
  <Issue_and_decision />
  <Type>Capability</Type>
  <Existential>Mandatory</Existential>
  - <Relationship type="Composed-Of">
    <Rel_feature>Infrared</Rel_feature>
    <Rel_feature>Bluetooth</Rel_feature>
    <Rel_feature>WAP</Rel_feature>
  </Relationship>
</Feature>
- <Feature>
  <Feature_name>Infrared</Feature_name>
  <Description>Via infrared customers can send or receive data such as business cards and calendar notes to and from a compatible phone or data device. Do not point IR infrared beam at anyone’s eye or allow it to interfere with other IR devices. The infrared ports of the sending and the receiving devices are pointing at the sending device and there are no obstructions between the devices. The preferable distance between two devices is one metre at most.</Description>
  <Issue_and_decision />
  <Type>Capability</Type>
  <Existential>Optional</Existential>
  - <Relationship type="Composed-Of">
    <Rel_feature>Sending Data Via Infrared</Rel_feature>
    <Rel_feature>Receiving Data Via Infrared</Rel_feature>
  </Relationship>
</Feature>
- <Feature>
  <Feature_name>WAP</Feature_name>
  <Description>The phones must be switched on to use this function. Do not switch the phone on when wireless phone use is prohibited or when it may cause interference or danger. Various WAP Wireless Application Protocol service providers on the Internet maintain pages specially designed for mobile phones offering services such as news weather reports banking travel information entertainment and games. These pages use the Wireless Markup Language WML Web pages using the Hypertext Markup Language HTML can not be viewed on the phones. Once the customers have stored all the required connection setting they can access WAP pages. There are three different ways to access WAP pages the homepage of customers’ service provider the bookmark from the Bookmarks view and the address of a WAP service.</Description>
  <Issue_and_decision/>
  <Type>Capability</Type>
  <Existential>Optional</Existential>
- <Relationship type="Composed-Of">
  <Rel_feature>Receiving Smart Messages</Rel_feature>
  <Rel_feature>Adding Bookmark</Rel_feature>
  <Rel_feature>Viewing Bookmark</Rel_feature>
  <Rel_feature>Sending Bookmark</Rel_feature>
  <Rel_feature>Browsing WAP Pages</Rel_feature>
  <Rel_feature>Saving WAP Pages</Rel_feature>
  <Rel_feature>WAP Connection Security</Rel_feature>
</Relationship>
</Feature>
- <Feature>
  <Feature_name>Sending Data Via Bluetooth</Feature_name>
  <Description>There can be only one active Bluetooth connection at a time. Data can be various depending on an application where the item the customers wish to send is stored. For example to send a photo to another device open the Images application. After opening the application the customers can select options of sending data via Bluetooth. The phone starts to search for devices within range Bluetooth enables devices that are within range start to appear on the display one by one. In the case that the customers have searched for Bluetooth devices earlier a list of the devices that were found previously is shown first. The phone is able to start and stop searching of Bluetooth devices before sending data. When the connection has been successfully established the phone is ready to send data If sending fails the message or data will be deleted.</Description>
  <Issue_and_decision/>
  <Type>Capability</Type>
  <Existential>Mandatory</Existential>
  <Relationship/>
</Feature>
<Feature_name>Receiving Data Via Bluetooth</Feature_name>
<Description>When the phone has received data via Bluetooth there will be a tone sound and a pop-up to ask if the customer want to accept the Bluetooth message. If accept the item will be placed in the Inbox folder in Messaging.</Description>

- <Feature>
  <Feature_name>Sending Data Via Infrared</Feature_name>
  <Description>The sending device has to select the desired infrared function to start data transfer. If data transfer is not started within one minute after the activation of the infrared port the connection is cancelled and must be restarted again.</Description>

- <Feature>
  <Feature_name>Receiving data Via Infrared</Feature_name>
  <Description>All items which are received via infrared are placed in the Inbox folder in Messaging. The receiving device must activate the infrared port.</Description>

- <Feature>
  <Feature_name>Receiving Smart Messages</Feature_name>
  <Description>The short message a so-called smart message can be received from the network operator or service provider that offers the WAP service. The message can contain both WAP access point settings and bookmarks such that the customers can view the bookmark and access point information separately.</Description>
<Feature>
  <Feature_name>Adding Bookmark</Feature_name>
  <Description>The customer can add a bookmark in the Bookmark view by only defining the address.</Description>
</Feature>

- <Feature>
  <Feature_name>Viewing Bookmark</Feature_name>
  <Description>Viewing Bookmark A bookmark consists of an Internet address mandatory bookmark title WAP access point and if the service requires a user name and password. The phone may have some pre-installed bookmarks for sites not affiliated with the phone company. The company does not warrant or endorse these sites. If the customer choose to access them the customer should take the same precautions for security or content as would with any site. In the Bookmarks view the customer can see bookmarks pointing to different kinds of WAP pages Bookmarks are indicated by the following icons.</Description>
</Feature>

- <Feature>
  <Feature_name>Browsing WAP Pages</Feature_name>
  <Description>The phone can show which WAP pages are previously visited. The customers can open a link with various input methods e.g. pressing the joystick entering addresses of WAP pages etc. then they can read and view WAP service messages while browsing.</Description>
</Feature>

- <Feature>
  <Feature_name>Saving WAP Pages</Feature_name>
  <Description>A WAP page can be saved to the phone memory and be viewed it offline. The customers can open the Saved pages view from the phone memory. Moreover the customer can start a connection to the WAP service and to retrieve the page again.</Description>
</Feature>
<Feature>
  <Feature_name>WAP Connection Security</Feature_name>
  <Description>The phone can pop-up an indicator during a WAP connection when the data transmission between the phone and the WAP gateway or WAP service is encrypted and secure. It is up to the service provider to secure data transmission between the gateway and the content server. It is possible that the customer is trying to access or have accessed confidential information requiring passwords for example the customer’s bank account. The phone can empty the cache after each use.</Description>
</Feature>

- <Feature>
  <Feature_name>Tools</Feature_name>
  <Description>Tools</Description>
</Feature>

- <Relationship type="Composed-Of">
  <Rel_feature>Calendar</Rel_feature>
  <Rel_feature>Games</Rel_feature>
  <Rel_feature>Clock</Rel_feature>
  <Rel_feature>Digital Camera</Rel_feature>
</Relationship>

- <Feature>
  <Feature_name>Clock</Feature_name>
  <Description>The phone can set and show the time and date. The clock can be Analogue or Digital according to the models of phones.</Description>
</Feature>

- <Relationship type="Composed-Of">
  <Rel_feature>A Call</Rel_feature>
  <Rel_feature>Wallpaper</Rel_feature>
  <Rel_feature>Data Transfer Setting</Rel_feature>
  <Rel_feature>Clock Setting</Rel_feature>
</Relationship>
<Feature><Feature_name>Screen saver</Feature_name><Description>The phone can set what is shown on the wallpaper.</Description><Issue_and_decision/>
<Type>Capability</Type><Existential>Optional</Existential></Feature>

- <Feature>
  <Feature_name>Wallpaper</Feature_name>
  <Description>The phone can set what is shown on the wallpaper.</Description>
  <Issue_and_decision/>
  <Type>Capability</Type>
  <Existential>Optional</Existential>
</Feature>

- <Feature>
  <Feature_name>Screen saver</Feature_name>
  <Description>The phone can set what is shown on the screen saver bar time and date or text.</Description>
  <Issue_and_decision/>
  <Type>Capability</Type>
  <Existential>Optional</Existential>
</Feature>

- <Feature>
  <Feature_name>Calling</Feature_name>
  <Description>The phone can connect to landline and mobile phones. During a call the customer can mute unmute end active call end all calls hold/unhold make a new call. Moreover if the phone is activated the Call waiting service the network will notify a new incoming call while the customer has a call in progress.</Description>
  <Issue_and_decision/>
  <Type>Capability</Type>
  <Existential>Mandatory</Existential>
</Feature>

- <Relationship type="Composed-Of">
  <Rel_feature>Making Call</Rel_feature>
  <Rel_feature>Receiving A Call</Rel_feature>
</Relationship>

- <Feature>
  <Feature_name>Making Call</Feature_name>
  <Description>The phone can make a call by entering the phone number via keypad or selecting a contact from the Contacts directory.</Description>
  <Issue_and_decision/>
  <Type>Capability</Type>
  <Existential>Mandatory</Existential>
</Feature>

- <Feature>
  <Feature_name>Receiving A Call</Feature_name>
  <Description>The phone can answer an incoming call by any input methods. The customers can ignore to answer a call by activating as a line busy. The customers can quickly mute the ringing tone for a coming call. Moreover the customer can
direct incoming calls to another phone number. This depends on the network service 
activated.</Description>

<Type>Capability</Type>
<Existential>Mandatory</Existential>
</Feature>

- <Feature>
  <Feature_name>Input Method</Feature_name>
  <Description>The phone has different ways for entering commands.</Description>

  <Type>Capability</Type>
  <Existential>Mandatory</Existential>
</Feature>

- <Relationship type="Composed-Of">
  <Rel_feature>Keypad</Rel_feature>
  <Rel_feature>Joy Stick</Rel_feature>
</Relationship>

- <Feature>
  <Feature_name>Keypad</Feature_name>
  <Description>A customer can enter data and activate with responses via keypad of the phone.</Description>

  <Type>Capability</Type>
  <Existential>Mandatory</Existential>
</Feature>

- <Feature>
  <Feature_name>Joy Stick</Feature_name>
  <Description>A customer can browse and select items with a joy stick which can move four directions left right down and up.</Description>

  <Type>Capability</Type>
  <Existential>Optional</Existential>
</Feature>

- <Feature>
  <Feature_name>Output Method</Feature_name>
  <Description>The phone has a screen as an output showing activating responses to a customer.</Description>

  <Type>Capability</Type>
  <Existential>Mandatory</Existential>
</Feature>

- <Relationship type="Composed-Of">
  <Rel_feature>Graphic display</Rel_feature>
  <Rel_feature>Text display</Rel_feature>
</Relationship>
Appendix C: Mobile Phone Product Line Software System Case Study

- <Feature>
  <Feature_name>Graphic display</Feature_name>
  <Description>The screen can display graphic mode including textual contents. The customer can change the contrast of the display to lighter or darker and the colour palette used on the display. The phone has the light sensor to measure the surrounding light. When the light sensor is active and it is bright enough the phone display and keypad lights are automatically shut down. The phone can control the setting of Minimum and Maximum of the sensitivity of the light sensor. Also the function can be set off if the customer does not want to use the light sensor. Moreover the display and keypad lights will shut down if there have been not key presses within a range of time depending on the models of the phones.</Description>
  <Issue_and_decision />
  <Type>Capability</Type>
  <Existential>Alternative</Existential>
  - <Relationship type="Composed-Of"> <Rel_feature>VGA</Rel_feature> <Rel_feature>"G"</Rel_feature> </Relationship>
</Feature>

- <Feature>
  <Feature_name>Text display</Feature_name>
  <Description>The screen can display responses in textual. The phone has the light sensor to measure the surrounding light. When the light sensor is active and it is bright enough the phone display and keypad lights are automatically shut down. The function can be set off if the customer does not want to use the light sensor. Moreover the display and keypad lights will shut down if there have been not key presses within a range of time depending on the models of the phones.</Description>
  <Issue_and_decision />
  <Type>Capability</Type>
  <Existential>Alternative</Existential>
</Feature>

- <Feature>
  <Feature_name>Data</Feature_name>
  <Description>Data</Description>
  <Issue_and_decision />
  <Type>Capability</Type>
  <Existential>Mandatory</Existential>
  - <Relationship type="Composed-Of"> <Rel_feature>Ring Tones</Rel_feature> <Rel_feature>Photo albums</Rel_feature> <Rel_feature>Phone Book</Rel_features>
<Rel_feature>Text Messages</Rel_feature>
<Rel_feature>Emails</Rel_feature>
<Rel_feature>Voice</Rel_feature>
</Relationship>
</Feature>
- <Feature>
  <Feature_name>Photo albums</Feature_name>
  <Description>Photo albums</Description>
  <Issue_and_decision />
  <Type>Capability</Type>
  <Existential>Optional</Existential>
  - <Relationship type="Composed-Of">
    <Rel_feature>photos</Rel_feature>
  </Relationship>
</Feature>
- <Feature>
  <Feature_name>photos</Feature_name>
  <Description>The phone must be switched on to use this function. The phone can view photos organise delete and send photos and pictures stored. There are different types of photos Standard or Night or Portrait modes. Photos in Standard or Night mode are saved in 640×480 pixel VGA format photos. In Portrait mode are saved in 80×96 pixel format. The phone can store view organise delete and send photos and pictures stored in your phone. The possible formats of photos in the phone can be JPEG GIF TIFF MBM BMP WBMP OTA WMP Unsupported or Unknown.</Description>
  <Issue_and_decision />
  <Type>Capability</Type>
  <Existential>Mandatory</Existential>
  - <Relationship type="Composed-Of">
    <Rel_feature>Storing Photos</Rel_feature>
    <Rel_feature>Viewing Photos</Rel_feature>
    <Rel_feature>Sending Photos</Rel_feature>
  </Relationship>
</Feature>
- <Feature>
  <Feature_name>Storing Photos</Feature_name>
  <Description>The phone can store photos including ones taken with the integrated digital camera in the phone itself or ones sent into inbox in a multimedia or a photo messaging as an e-mail attachment via an infrared or Bluetooth connection.</Description>
  <Issue_and_decision />
  <Type>Capability</Type>
  <Existential>Mandatory</Existential>
</Feature>
- <Feature>
  <Feature_name>Viewing photos</Feature_name>
  <Description>The phone can view an photo with functions e.g. zooming full screen moving the focus rotating.</Description>
  <Issue_and_decision/>
  <Type>Capability</Type>
  <Existential>Mandatory</Existential>
</Feature>
- <Feature>
  <Feature_name>Sending photos</Feature_name>
  <Description>The phone can send photos in a multimedia or e-mail message as an attachment or communication ports like infrared Bluetooth.</Description>
  <Issue_and_decision/>
  <Type>Capability</Type>
  <Existential>Mandatory</Existential>
</Feature>
- <Feature>
  <Feature_name>Text Messages</Feature_name>
  <Description>The phone can edit, send and receive a short message photos in a multimedia or e-mail message as an attachment or communication ports like infrared Bluetooth.</Description>
  <Issue_and_decision/>
  <Type>Capability</Type>
  <Existential>Mandatory</Existential>
</Feature>
- <Relationship type="Composed-Of">
  <Rel_feature>Sending Text Message</Rel_feature>
  <Rel_feature>Receiving Text Message</Rel_feature>
  <Rel_feature>Edit Text Message</Rel_feature>
</Relationship>
- <Feature>
  <Feature_name>Sending Text Message</Feature_name>
  <Description>The phone can send a short message to another phone by pressing a contact via keypad or entering from a list of contacts. The maximum amount of sending short messages and the maximum amount of characters in one short message depend on the models of the phone. The phone can store messages sent previously in the outbox.</Description>
  <Issue_and_decision/>
  <Type>Capability</Type>
  <Existential>Mandatory</Existential>
</Feature>
- <Feature>
  <Feature_name>Receiving Text Message</Feature_name>
  <Description>The phone can receive a short message and store it in the phone memory. The maximum amount of receiving short messages and the maximum
amount of characters in one short message depend on the models of the phone. The phone can store messages received previously in the inbox.

```xml
<Type>Capability</Type>
<Existential>Mandatory</Existential>
</Feature>
- <Feature>
  <Feature_name>Phone Book</Feature_name>
  <Description>PhoneBook</Description>
  <Issue_and_decision />
  <Type>Capability</Type>
  <Existential>Mandatory</Existential>
  - <Relationship type="Composed-Of">
    <Rel_feature>Contacts</Rel_feature>
  </Relationship>
</Feature>
- <Feature>
  <Feature_name>Contacts</Feature_name>
  <Description />
  <Issue_and_decision />
  <Type>Capability</Type>
  <Existential>Mandatory</Existential>
</Feature>
- <Feature>
  <Feature_name>Emails</Feature_name>
  <Description>The phone can send receive retrieve reply to and forward email. To do so the customer must configure an Internet Access Point IAP correctly and define the email settings correctly. Emails can be attached with photos.</Description>
  <Issue_and_decision />
  <Type>Capability</Type>
  <Existential>Optional</Existential>
  - <Relationship type="Composed-Of">
    <Rel_feature>Sending Emails</Rel_feature>
    <Rel_feature>Receiving Emails</Rel_feature>
  </Relationship>
</Feature>
- <Feature>
  <Feature_name>Sending Emails</Feature_name>
  <Description>The customers can select the recipient(s) from a list of contacts in the phone or write the email address of the recipient. It can send emails to multiple recipients. Moreover the customer can add an attachment to an e-mail. The attachment includes photos sound clip note etc. This function includes replying and forwarding emails.</Description>
  <Issue_and_decision />
  <Type>Capability</Type>
```
<Existential>Mandatory</Existential>
</Feature>
- <Feature>
  <Feature_name>Retrieving Emails</Feature_name>
  <Description>The customers can receive emails and store them in the inbox emails. Afterwards the phone can retrieve and delete those emails.</Description>
  <Issue_and_decision/>
  <Type>Capability</Type>
  <Existential>Mandatory</Existential>
</Feature>
</Feature_Model>

References

Appendix D: The Center of Excellence for Software Traceability

The International Center of Excellence for Software Traceability (CoEST) was established in 2005 with the charge to “bring together traceability researchers and experts in the field, encourage research collaborations, assemble a body of knowledge for traceability, and develop new technology to meet tracing needs.”

CoEST membership currently includes academic, government, and industrial researchers from across the U.S. and Europe.

Since its inception, the CoEST has engaged in two primary projects, the Grand Challenges of Traceability (GCT) and the Tracy project. The grand challenges, which were presented in chapter “The Grand Challenge of Traceability (v1.0)” of this book, provide a detailed road map of critical research and practice goals. The Tracy project is driven by the grand challenges, and as depicted in Fig. 1, focuses on...
equipping the traceability research community through building research infrastructure, collecting and organising datasets, establishing benchmarks, and developing a tool named TraceLab, which will provide support for designing and executing a broad range of traceability experiments.

For further information, or to become a CoEST member, please visit http://www.coest.org.
Appendix E: TraceLab – A Tool for Supporting Traceability Research

TraceLab is a visual experimental workbench for designing and executing traceability experiments. TraceLab’s primary features include:

- A visual environment for designing and executing experiments.
- A component library which facilitates sharing a wide variety of importers, pre-processors, algorithms, analysers, etc. across the traceability community.
- Ability to write components in a wide variety of languages including, C++, C#, and Java, and combined into a single experimental workflow.
- Integration with standard benchmarks for comparatively evaluating new techniques against previous results. Benchmarks utilise community datasets, standardised metrics, and previously published traceability tasks.
- A scalable environment that supports experiments involving extremely large sized industrial datasets.
- Portability across multiple operating systems including Windows, Linux, and Mac OS.
Appendix E: TraceLab – A Tool for Supporting Traceability Research

- A simple installation process which allows new users to quickly download and install TraceLab.
- An intuitive user interface which enables new users to execute basic experiments without any formal training.

The TraceLab project is funded under National Science Foundation’s Major Research Instrumentation Grant # 0959924. Further information is available at http://www.coest.org.
Index

A
Actor, 159, 245, 454–457, 464
Agility, 265–268
Architecture
architect, 51, 129, 313
Aspect oriented
design, 281
programming, 278, 312

B
Benchmark, 357, 367, 377, 396–397, 406
Business case, 54–57, 65

C
Challenge
configurable, 367–373
cost, 43, 69, 183, 344
effective, 183
portable, 288–393
purposed, 345, 349–360, 389, 401, 403, 405, 407
scalable, 345, 349, 381–388, 402–403, 405, 408
trusted, 373–381
ubiquitous, 399–404
valued, 345, 349–351, 393–399, 401, 405, 408
Change
evolution, 17, 174, 181, 209
impact, 4, 28, 32, 52, 178, 241, 268, 301, 346
analysis, 4, 32, 178, 241, 268, 301
management, 13, 17, 35–38, 270, 323, 346
Clustering, 80–81
CoEST (Center of Excellence for Software Traceability), 4, 343, 351, 405–406, 413
Configuration management, 142, 270–271, 325, 328, 403
Cost
benefit, 1, 12, 23–41, 190, 364, 366, 383, 387, 403
effort, 112
estimation, 29–33, 40
return on investment, 1, 24–25, 27, 29, 33, 37, 40, 55, 61, 112, 318, 361–364, 367, 395

D
Design
decision, 23, 38, 300–301, 303–304, 309, 311, 316
model, 183, 267, 312, 316
pattern, 309, 315, 431
Domain knowledge, 104, 106–107, 109, 114–115, 117, 247

E
Empirical study, 78, 90, 103, 114–122, 248–249, 386
Extreme Programming, 266–267
| F | Feature model, 149–150, 152–153, 156–158, 196, 205, 207, 218, 439, 442–444, 469–481 |

| G | Glossary, 413–424 |
| Goal oriented, 11, 175, 313 |
| Grand Challenge, 4, 183, 296–297, 341, 343–408 |

| H | Hazard, 324, 326, 332–334 |
| Health care, 425–437 |
| High assurance, 303–304 |
| Horizontal trace, 208 |
| Human analyst, 242, 245–246, 309 |
| Hypertext, 99, 102 |

| Infrastructure, 12, 62, 309, 313, 316, 334, 393 |

| J | Joinpoint, 279, 281, 287, 293–295 |

| K | Keyword, 133, 279, 288 |

| L | Latent Semantic Indexing (LSI), 73, 94, 99, 216 |

| Mandate, 37, 40, 47, 195, 241, 289, 331, 334, 352, 382, 393, 398 |
| Manual Tracing, 10, 91, 242–244, 253–254, 260, 415 |
| Medical device, 263, 321–337 |
| Meta model, 178, 183, 191, 196, 199–207, 289, 303, 312, 369 |
| Metrics |
| precision, 85–86, 101, 106 |
| average, 86, 92–93, 100, 118, 166 |
| recall, 375 |
| Missing element, 69, 148–149, 158, 161–162, 164, 166, 168 |
| Missing link, 107–108, 114, 123, 245, 253, 259 |
| Model |
| based, 80, 174, 177–178, 185, 187, 189–190, 196, 201–202, 272, 288 |
| Modeling language, 5, 48, 51, 175–177, 267, 310, 383 |
| Morphological, 75, 90 |
| Multimedia, 347, 358–386, 382, 444–447, 469, 478–479 |

| N | Natural language, 21, 74, 82, 108–110, 383, 453 |
| Non functional requirement, 36–37, 277, 299, 311, 324, 384, 387, 431, 437 |
O
Obsolete, 71, 140, 178, 184, 225, 228, 230, 378, 415, 420
OMG (Object management group), 177–178, 288, 439
Ontology, 248, 383
Organizational context, 268, 304

P
Policies, 182, 300, 347, 388–390, 392–393, 408
Post requirements traceability, 19, 385
Predictor, 251, 258–259
Pre requirements traceability, 59
Privacy, 279, 301, 425, 428
Process
assessment, 263, 322–323, 334
improvement, 53–55, 57, 59–60, 64, 66, 331, 344, 353, 358–359, 376, 396, 421
Product line
feature, 151
product requirements, 56, 198, 202, 209, 269, 352, 361
variability, 196, 200
Program comprehension, 71, 101, 104, 123
Project manager, 35, 94, 101, 124, 175, 200, 301, 331–332, 347, 395
Project size, 31, 104–105, 123, 273, 383, 427
Protocol, 159, 182, 206, 392, 442, 445, 453, 456
Prototype, 148, 164, 168, 187, 189, 347, 386, 391

Q
Quality
concern, 301–303, 305–308, 313–315, 379
goal, 301–302, 305, 313
non functional requirement, 36–37, 263, 277, 299–318, 324, 382, 384, 387, 426, 431, 437

R
Recovery technique, 73–75, 130, 143, 180, 384–385
Refactor, 36, 112, 183, 267, 426–427
Requirements
management, 38, 46, 416
prioritization, 88
specification, 5, 58, 78, 215, 301, 426
RETRO, 251–254
Risk
analysis, 334, 337
hazard, 332
management, 321, 324–326, 332, 334–335, 337
Roadmap, 62, 336, 406

S
Search engine, 114, 138, 140
Software Artifact, 71–75, 78, 89–90, 94, 151
Software evolution, 71, 201
Software maintenance, 104, 113, 115, 129, 325
Source artifact, 5–8, 14, 20, 414–419
Source code, 106–107, 111–113, 117–118, 432–433, 437
Staff turnover, 36, 270
Stakeholder, 56, 270, 346, 353, 384, 416
Stemmer, 75
Stop word, 74–75, 90, 245
Subsystem model, 153, 439, 445
Synonym, 81, 152, 157, 163
Systems traceability, 9, 45, 417

T
Tagger, 148–149
Index


Time stamp, 8, 417

Tooling
acquisition, 44, 52–55, 59, 61, 64–66
chain, 48, 50–51

Topic model, 78, 130

Trace
acquisition, 171, 195–211, 417
creation, 15, 26, 31–32, 40–41, 151, 166, 318, 364, 371, 382, 400, 403, 417, 419
element, 5–6, 8, 14, 375, 389, 391–392, 414, 417–418
evolution, 402
history, 388, 420
maintenance, 26, 31, 144, 350, 358, 371, 385, 400, 402–404, 418
obsoletion, 71, 140, 225, 228, 230, 415
planning, 1, 11, 25, 49, 350, 362, 368, 395, 401, 421–422
quality, 24, 26–29, 33, 37, 354, 418
query, 271, 418
record, 288, 418
recovery, 15, 108, 130, 143, 375, 384–385, 416–419, 421
relationship, 7, 13, 69, 130, 147–152, 154–156, 158–168, 311, 419–420
set, 16, 375, 381, 415, 418–420, 423
tool, 137

Traceability
acquisition, 171, 195–211, 417
analysis, 326–327, 333, 337, 343, 435–437
decay, 174, 179–180, 183–184, 378, 420
management, 11, 183, 332, 335, 421
planning, 11, 25, 49, 350, 362, 368, 395, 401, 421
recovery, 71–95, 99–124
rule, 33, 102, 147–150, 155–158, 160–168

Traceability body of knowledge (TBOK), 21, 355, 360, 367, 380–381, 398–399, 419

Traceability framework, 148–158

Traceability graph, 181, 303, 420

Traceability Information Model (TIM), 13, 49, 151–154, 159, 168, 171, 173, 176, 219–221, 226–227, 269, 272, 303, 368–373, 390, 393, 401, 407, 420

Traceability link
course-grained, 309, 316–317
link granularity, 5, 9, 11–12, 27, 32–33, 61, 112, 138, 142–144, 200–201, 304
source, 71, 73, 75, 78, 80, 88, 90, 101
target, 219, 226–228


Traceability network, 380, 388–389, 391–392, 421

Traceability practices, 21, 63, 197, 309, 328, 330, 332, 344, 355, 364, 384, 397, 421

Traceability Tools
ArchStudio, 130–131, 134–135
RETRO, 251–254
TraceLab, 364, 406, 411, 423, 485–486
Tracer, 10, 13, 157, 162, 164, 284, 416, 423
Trace recovery, 15, 108, 130, 143, 375, 384–385, 418
Trace relationship, 7, 13, 69, 130, 147–152, 154–156, 158–168, 311, 419
Track change, 201
Transaction, 30, 270, 273, 428, 433, 436–437

U
UML (Unified Modeling Language)
   class diagram, 5, 21, 150, 153, 176, 183, 215, 218, 439, 453, 459, 466–468
   sequence diagram, 153, 218, 439, 460–463, 467
Usage scenario, 130–132, 134, 138, 140

V
Variability model, 198–204, 207–210
Vector Space Model (VSM), 76, 87, 99, 119
Vendor, 50, 57, 62–63
Vertical trace, 18–19, 423
Version control, 182, 272–273, 379, 403, 418, 420, 423
Vision, 183, 341, 345–349, 352, 361, 368, 373–374, 381, 388, 394, 399

W
Workflow, 50, 57, 60, 356–357, 400, 485

X
Xpath, 158
Xquery, 147–148, 155–156, 162, 164, 168