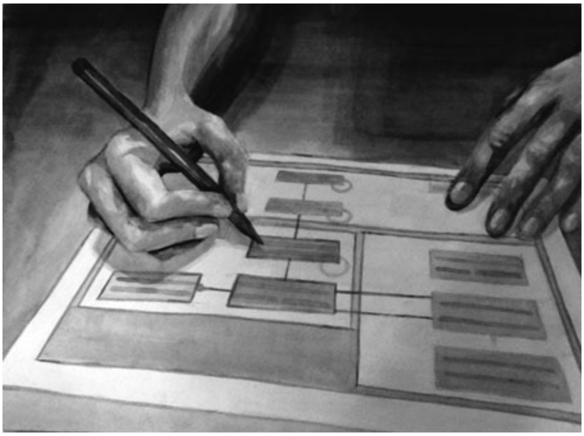

Part I
Traceability Strategy

Traceability needs to be planned for and managed if it is to be effective and remain effective in any particular project context. Stakeholders need to be identified and requirements determined. A suitable traceability process needs to be designed and potential support from tooling explored. However, all this initial effort is mute if there is no clear understanding of the anticipated return on investment from implementing traceability within an organisation. Traceability strategy comprises all those activities associated with traceability planning and traceability management.

In this first part of the book, the chapter "Traceability Fundamentals" defines a number of traceability-related terms and concepts, as they will be used throughout the remainder of the book. A simple process for analysing the cost-benefit of traceability and selecting a strategy accordingly is described in the chapter "Cost-Benefits of Traceability". A cautionary seven-step guide for making informed decisions about tool acquisition is presented in the chapter "Acquiring Tool Support for Traceability". In combination, the chapters "Cost-Benefits of Traceability" and "Acquiring Tool Support for Traceability" highlight important considerations to help plan and manage traceability in practice.

Traceability Fundamentals

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

1 Introduction

The role of traceability was recognised in the pioneering NATO working conference held in 1968 to discuss the problems of software engineering (Naur and Randell, 1969). One of the working papers in this conference examined the requirements for an effective methodology of computer system design and reported on the need to be able to ensure that a system being developed actually reflects its design. In a critique of three early projects focused on methodology, each was praised for the emphasis they placed on making "the system that they are designing contain explicit traces of the design process" (Randell, 1968).

Traceability was subsequently noted as a topic of interest in one of the earliest surveys on the state of the art and future trends in software engineering (Boehm, 1976), and its practice was certainly evident in those domains concerned with developing early tool support (Dorfman and Flynn, 1984; Pierce, 1978). By the 1980s, traceability could be found as a requirement in a large number of national and international standards for software and systems development, such as the high-profile DOD-STD-2167A (Dorfman and Thayer, 1990). Published research began to proliferate and diversify in the area of traceability in the late 1990s, spurred somewhat by renewed interest in the topic arising from two newly formed International Requirements Engineering professional colloquia, with two early papers focusing on the issues and problems associated with traceability (Ramesh and Edwards, 1993; Gotel and Finkelstein, 1994), the latter providing for the first systematic analysis of the traceability problem. The topic of traceability continues to receive growing research attention in the twenty-first century, with a particular focus on automated trace generation (Cleland-Huang et al., 2007; Hayes et al., 2006) and with concomitant advances in model-driven development (Aizenbud-Reshef et al., 2006; Galvao and Goknil, 2007; Winkler and von Pilgrim, 2010).

O. Gotel (✉)
New York, NY 10014, USA
e-mail: olly@gotel.net

J. Cleland-Huang et al. (eds.), *Software and Systems Traceability*,
DOI 10.1007/978-1-4471-2239-5_1,

However, despite the introduction of widely-available commercial tools claiming to support traceability in the 1980s, and substantive growth in this market through the 1990s and millennium, the actual practice of traceability remains poorly documented and, where it is examined (Mäder et al., 2009b), it appears to be little influenced by research. One confounding factor is inconsistency in the use of traceability terminology and concepts, not only between researchers and practitioners, but also within each of these communities themselves.

This chapter seeks to provide a resource on traceability fundamentals.[1] It defines the essential traceability terminology in Section 2 and is supplemented by an extensive glossary[2] that has been developed and endorsed by members of the traceability community. This glossary can be found as an appendix to this book and provides definitions for all the terms that are italicised in this chapter. The chapter also offers a model of a generic traceability process in Section 3 and describes the basic activities involved in the life cycle of a trace. This model is used as a frame of reference for articulating the grand challenge of traceability in the chapter by Gotel et al. of this book. Section 4 describes the basic types of traceability and explains some key associated concepts. Section 5 concludes the chapter.

2 Essential Traceability Terminology

At the most fundamental level, traceability is simply the potential to relate data that is stored within artifacts of some kind, along with the ability to examine this relationship. The ability to achieve traceability therefore depends upon the creation of navigable links between data held within artifacts that are otherwise disconnected. The value of traceability lies in the many software and systems engineering activities and tasks that the information provided through such interrelations can enable, such as change impact analysis, coverage analysis, dependency analysis, etc. (Gotel and Finkelstein, 1994; Lindvall and Sandahl, 1996; Ramesh and Jarke, 2001); tracing can provide visibility into required aspects of the software and systems development process and contribute to a better understanding of the software system under development.

This section defines two underlying terms, *trace artifact* and *trace link*, that are the building blocks of traceability. It subsequently uses these definitions to clarify the term *trace*. Based upon these definitions, the terms *traceability* and *tracing* are then defined.

[1] Section 3 of this chapter includes reproduced material from Center of Excellence for Software Traceability Technical Report #CoEST-2011-001, with permission. Please direct any feedback on this material via the CoEST website (http://www.coest.org).

[2] Version 1.0 of the traceability glossary is provided as an appendix to this book and the latest version of the glossary is maintained at http://www.coest.org. Please note that all glossary terms are defined using U.S. English.

2.1 Trace Artifact

Trace artifacts are traceable units of data. They refer to any residual data or marks of the software and systems development process that are made amenable to being traced. The term can apply to a single requirement, a cluster of requirements, or even to an entire requirements specification document. The term can apply to a Unified Modeling Language (UML) class diagram, a single class therein, or even to a particular class operation. For conceptual simplicity, the general term "artifact" is used to apply to both the object as a whole and to any internal delineation therein. What this means is that the granularity of a trace artifact is not pre-determined and may not even be consistent in any one particular project. It is this uncertainty in the granularity of trace artifacts that can lead to many problems in establishing and using traceability in practice.

> **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*.

Three terms closely associated with trace artifact include *trace artifact type*, *source artifact* and *target artifact*. The trace artifact type serves to classify the nature and function of the artifact, and is usually a recognised and "documented" by-product of the software and systems development process. The terms source artifact and target artifact serve to characterise the role of a particular trace artifact in a specified 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*.
>
> **Source artifact** – The *artifact* from which a *trace* originates.
>
> **Target artifact** – The *artifact* at the destination of a *trace*.

2.2 Trace Link

A *trace link* is a single association forged between two trace artifacts, one comprising the source artifact and one comprising the target artifact. This definition of trace link implies that the link has a primary direction for tracing, from the source artifact to the target artifact. Directionality between the two trace artifacts provides for the ability to traverse the trace link, or to follow it, so as to associate the two

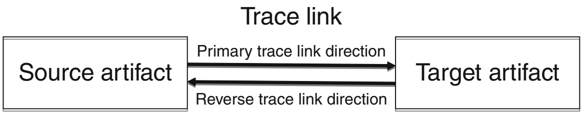

Fig. 1 Trace link directionality

pieces of data. It is this juxtaposition that is sought through traceability, rather than the pure retrieval of one piece of data. In practice, however, every trace link can be traversed in two directions, so the trace link also has a reverse trace link direction and is effectively bidirectional, as illustrated in Fig. 1.

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*.

The directionality of a trace link is therefore an important concept. Where a source artifact and a target artifact are defined, the semantics of the directionality is clear. Whether or not the trace link can physically be navigated in both directions, however, is usually a matter of implementation. Three terms clarify the directionality inherent in a trace link, the *primary trace link directio*n, the *reverse trace link direction* and the concept of a *bidirectional trace link*.

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).

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).

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*.

Two interrelated terms that are closely associated with trace link are *trace link type* and *link semantics*. The trace link type serves to classify the nature and function of the trace link. It is usually characterised according to the meaning of the relationship between the two artifacts that the link associates, so the trace link type is generally defined in terms of the link's semantic role. The trace link type is a broader term that may define a collection of links with the same link semantics.

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*.

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.

The term *trace relation* is frequently used interchangeably with the term trace link in many publications. In reviewing the traceability fundamentals and encouraging the more consensual use of terminology within the traceability community, the proposal is to differentiate the two terms in the future. Following from database theory, a trace relation describes all the trace links that are specified between two defined artifact types acting as source artifacts and target artifacts. It is the trace relation that is captured in the commonly used *traceability matrix*.

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*.

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

2.3 Trace

Use of the term *trace* has led to some misunderstanding in the traceability community since it has two distinct meanings dependent upon whether the term is being used as a noun (i.e., "a mark remaining" (OED, 2007)) or as a verb, (i.e., "tracking or following" (OED, 2007)). When used in a software and systems engineering context, the meanings are often used interchangeably whereas they need to be distinguished. "Trace" can, therefore, be defined in two ways.

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*).

When used as a noun, the term "trace" refers to the complete triplet of *trace elements* that enable the juxtaposition of two pieces of data: the source artifact, the target artifact and the trace link. Additional information, in the form of *trace attributes*, may qualify properties of the overall trace or of each of the three elements. Such traces can either be *atomic* or *chained* (see Fig. 2). Where chained, the trace links are strung together by the source and the target trace artifacts that they connect, the target artifact for one trace becoming the source artifact for the subsequent trace, to form a series of data juxtapositions.

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

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.*

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 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*.

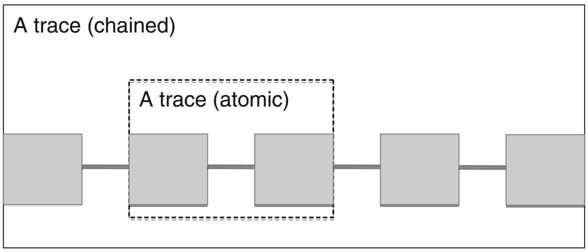

Fig. 2 A trace provided via a single trace link or via a chain of trace links

When used as a verb, the term "trace" (i.e., to trace) is associated with the activity of tracing (see Section 2.5).

2.4 Traceability

Traceability is the potential for traces (as defined above in the noun sense) to be established (i.e., created and maintained) and used. The challenge for traceability is that each of the component elements (i.e., the trace artifacts and trace links) needs to be acquired, represented and stored, and then subsequently retrieved as a trace to enable software and systems engineering activities and tasks. Both the time and the manner in which traces are established and brought together for use will depend upon the purposes to which the traceability is put. Consequently, traces exist within their own life cycles and can (ideally) be reused in different contexts. The type and the granularity of the trace artifacts, and the semantics of the trace link, are therefore details that are best determined on a project-by-project basis. They could perhaps even be determined on a moment-to-moment basis in relation to an overarching traceability strategy. It is this process through which traces come into existence and eventually expire that influences the definition of a generic traceability process model in Section 3.

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*.

Frequently used terms include requirements traceability, software traceability and systems traceability. These all delineate the artifact types that are the primary objects of interest for tracing purposes. For example, in the case of requirements traceability, this focuses explicitly on the potential to establish and use traces that associate requirements-related artifacts in some way or another. Other more specific traceability terms are defined in the glossary that accompanies this book.

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.)

2.5 Tracing

Tracing implies undertaking all those activities required to put traceability in place, in addition to all those activities that exploit the results.

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

Tracing activities demand some form of agency, and leads to the three associated terms of *manual*, *automated* and *semi-automated tracing* when referring to the nature of the activity that puts the traceability in place.

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*.

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.

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.

3 A Generic Traceability Process Model

Figure 3 depicts a generic *traceability process model*. It shows the essential activities that are required to bring traces into existence and to take them through to eventual retirement. Traces are created, maintained and used, all within the context of a broader traceability strategy. This strategy provides the detail of stakeholders' needs, decisions regarding mechanism and automation, and also chains atomic traces in some agreed way to enable required activities and tasks. Continuous feedback is a critical aspect of the entire process to enable the traceability strategy to evolve over time. The four key activities of this generic traceability process model are described in the following sub-sections.

Traceability process model – An abstract description of the series of activities that serve to establish *traceability* and render it usable, along with a

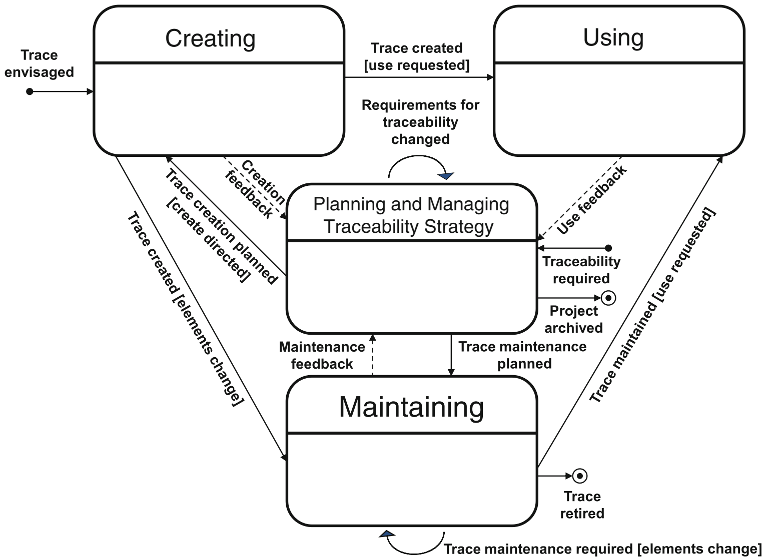

Fig. 3 A generic traceability process model

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*.

3.1 Traceability Strategy

Effective traceability rarely happens by chance or through ad hoc efforts. Minimally, it requires having retained the artifacts to be traced, having the capacity to establish meaningful links between these artifacts and having procedures to interrogate the resulting traces in a goal-oriented manner. Such simple requirements conceal complex decisions as to the granularity, categorisation and storage of assorted multi-media artifacts. It also conceals choices as to the approach for generating, classifying, representing and then maintaining their inter-artifact and intra-artifact linkages. Additional questions need to be answered, such as: Which of these tracing activities should be manual? Which should be automated? Where should the responsibilities for these activities lie? When should they be undertaken? There are many decisions that need to be made and, therefore, an enabling traceability strategy needs to be built into the engineering and management practices from day one on a software and systems engineering project. Figure 4 outlines the typical high-level activities associated with planning and managing a traceability strategy.

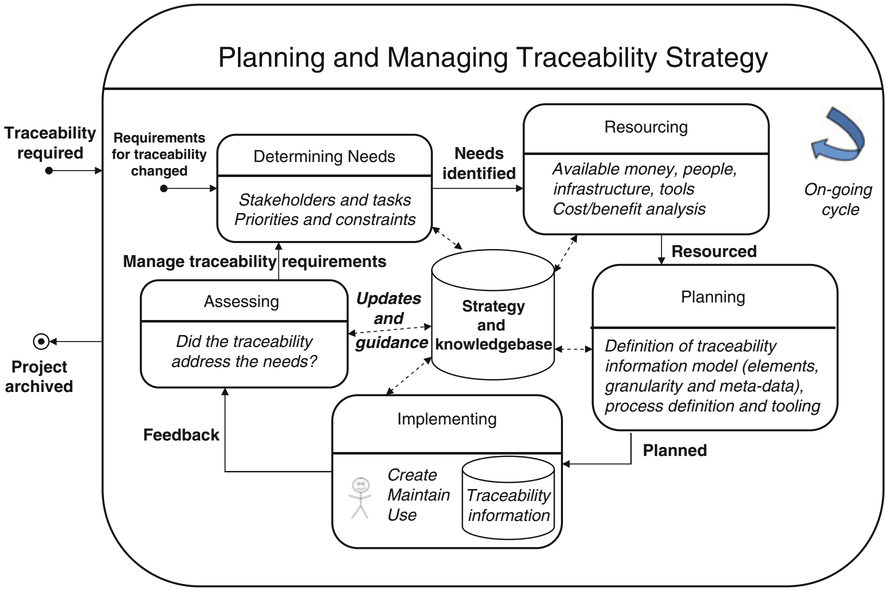

Fig. 4 Planning and managing a traceability strategy

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 is concerned with the provisioning of information to help in answering project-specific questions and in undertaking project-directed activities and tasks; it is thus a supporting system rather than a goal in its own right. This perspective demands understanding those stakeholders who may need the potential for traceability, what for and when? Acquiring clear-cut answers to these questions at the start of a project is not straightforward, as both stakeholders and their task needs will change. Even if these could be articulated exhaustively, building a *traceability solution* to service all needs is unlikely to be cost-effective, as resources are generally limited in some finite way. Determining whose needs to satisfy, and so which *traceability-enabled activities and tasks* to facilitate, is a value decision that lies at the heart of a traceability strategy; determining needs and resourcing constraints is a precursor to any discussion about trace artifacts, trace links and mechanism.

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 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., 2009a) for more detail.)

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 tool – Any instrument or device that serves to assist or automate any part of the *traceability process*.

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.

Ensuring that the traceability is then established as planned, and yet can adapt to remain effective as needs evolve and as a project's artifacts change, is also the province of traceability strategy. Determining how the traceability will be provisioned such that the requisite quality can be continuously assured further demands analysis, assessment and potential modification of the current traceability solution. Assessing the quality and the execution of the traceability solution, and implementing a feedback loop to improve it, is a critical part of the traceability strategy for a project; it needs to develop and leverage historical *traceability information*.

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

Within the context of a broader traceability strategy, the creation, maintenance and use of individual traces and their constituent elements all need to be defined and managed. Given that atomic traces comprise source, target and relational elements, these data requirements need to be identified. This includes decisions as to meta-data to associate, dependent upon what kinds of traceability-enabled activities and tasks the trace is anticipated to participate in and support. Resourcing, planning and implementation decisions may hence vary on a trace-by-trace basis; for instance, it is quite possible that a particular trace is not created or maintained until its use is actually required. Traces thereby inhabit independent life cycles, the constituent activities of which are examined in the following sections.

3.2 Traceability Creation

When creating a trace, the elements of the trace have to be acquired, represented and then stored in some way, as illustrated in Fig. 5. Reference models and classification schemes characterising different types of trace link and trace artifacts drive the *traceability creation* process, as usually defined within the traceability information model of the overarching traceability strategy.

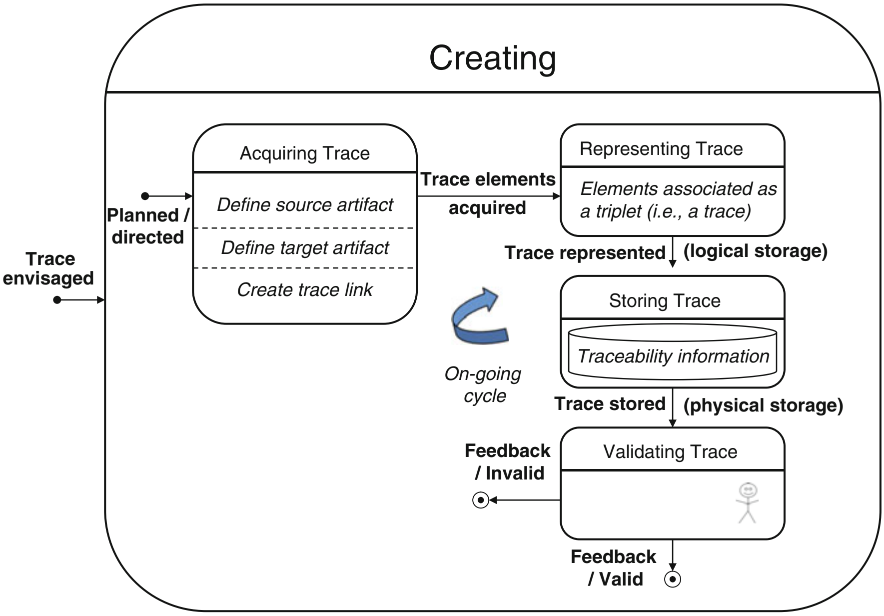

Fig. 5 Traceability creation

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*.

While project artifacts are generally pre-existing on a project, the links between them may not yet be defined. Techniques to support the creation of trace links can range from manual to automated approaches, each with differing degrees of efficiency and effectiveness. The differentiating factor is often whether the trace links are created concurrently with the forward engineering process (i.e., *trace capture*) or at some point later (i.e., *trace recovery*). Validation is therefore critical to the viability of the traceability creation process, regardless of how trace links are initially created, as it is concerned with determining and assuring the credibility of the trace as a whole.

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 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.

3.3 Traceability Maintenance

An association made between two artifacts at a moment in time to serve a particular purpose does not automatically mean that the resulting trace will have a persistent, useful life. The need for maintenance on a trace can be triggered by changes to any of the trace's elements that, in turn, can be triggered by changes to elements within a chain. *Traceability maintenance* can also be required following changes to the requirements and constraints that drive the overarching traceability strategy.

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.

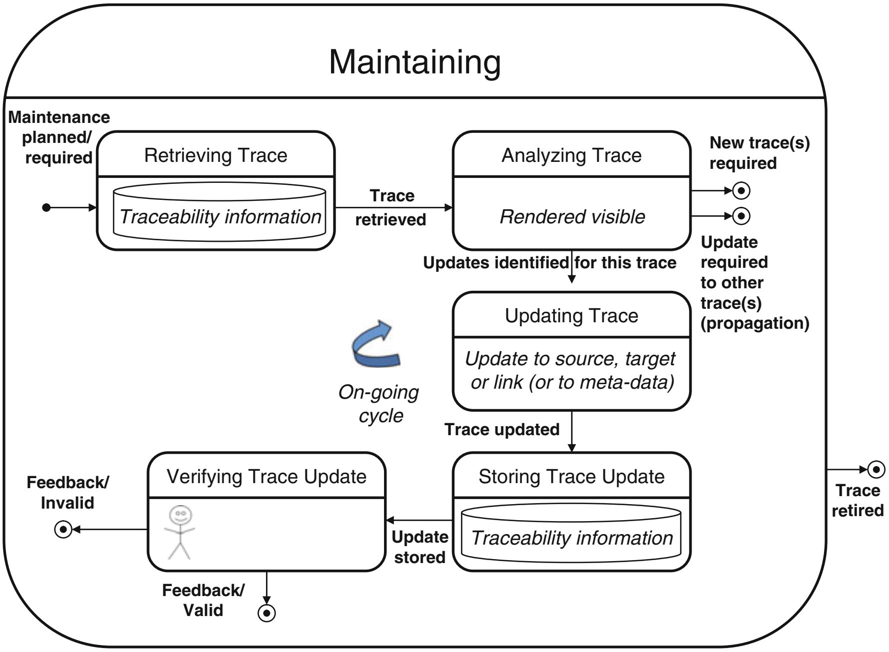

Fig. 6 Traceability maintenance

To maintain a trace, it needs to be retrieved and the nature of the change analysed to determine what update is necessary, as illustrated in Fig. 6. This may necessitate the propagation of changes and/or the creation of entirely new traces. Updates need to be performed, where applicable, recorded and verified. Feedback on the maintenance process is also essential for evolving the overarching traceability strategy. As per traceability creation, traces can be maintained continuously or on-demand.

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

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*.

3.4 Traceability Use

The availability and usefulness of traces has to be ensured to allow for their ongoing use throughout the software and systems development life cycle, potentially in a myriad of configurable ways. Here, it is helpful to distinguish between short-term *traceability use* during initial product development and long-term traceability use during subsequent product maintenance. Typical short-term uses for traceability include requirements completeness analysis, requirements trade-off analysis or requirements-to-acceptance-test mapping for final acceptance testing. Typical examples of long-term uses for traceability include the determination of effects of changes to a software system or the propagation of changes during its evolution.

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.

Any atomic trace is likely to play a role in the context of many use contexts. To use a trace in isolation, or as a constituent part of a chain, it needs to be retrieved and rendered visible in some task-specific way, as suggested in Fig. 7. An important component of the use process is assessing the quality of the traceability that is provided in terms of the fitness for purpose with respect to the task or activity for which the traceability is required. Such information provides a feedback loop to improve the overall traceability strategy.

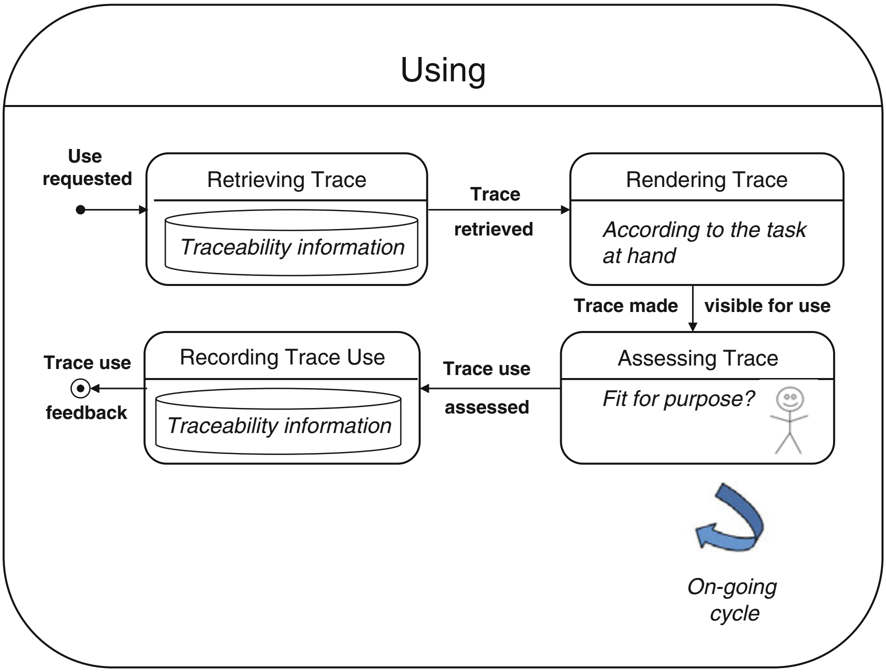

Fig. 7 Traceability use

4 Basic Types of Traceability and Associated Concepts

Additional terms that delineate different basic types of traceability are highlighted in the context of Fig. 8 and defined below.

The traceability of Fig. 8 is bidirectional. *Forward traceability* offers the potential to link a single requirement statement to those methods of the class designed to implement it, and subsequently to follow this trace link to reveal the forward engineering process. *Backward traceability* offers the potential to link the class methods back to the requirement that they help to satisfy, and subsequently to follow this trace link to reveal the reverse engineering process. The forward and the backward direction pertain to the logical flow of the software and systems development process. These are the fundamental and primitive types of tracing.

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*.

Backward traceability – The potential for *backward tracing.*

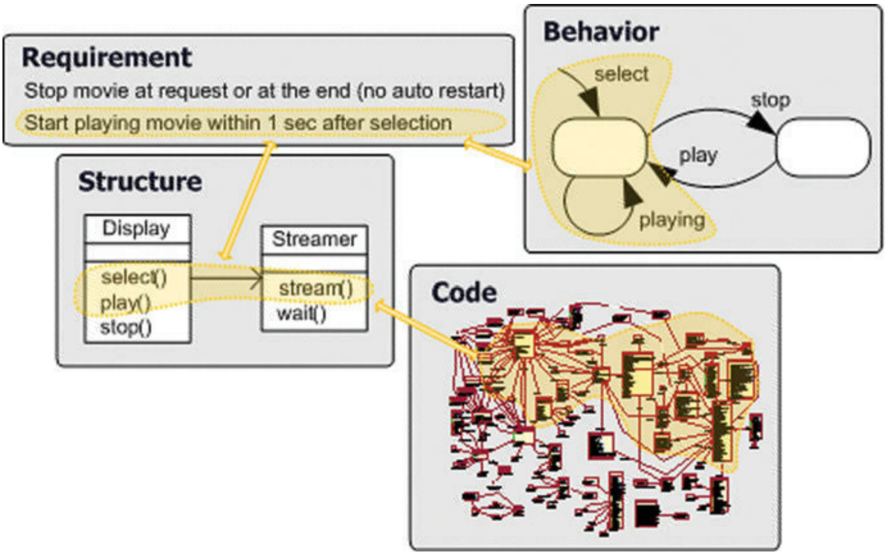

Fig. 8 A simplified, but typical, tracing context

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*.

In Fig. 8, the potential to trace from the requirement through to the code is *vertical traceability*, linking artifacts at differing levels of abstraction to accommodate life cycle-wide or end-to-end traceability. Any potential to trace between versions of the requirement or versions of the code is *horizontal traceability*, linking artifacts at the same level of abstraction at different moments in time to accommodate versioning and rollback. These two types of tracing, vertical and horizontal, employ both forward and backward tracing.

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*.

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*.

Two additional types of traceability are more conceptual in nature, and these can employ each of the above tracing types in some combination. *Post-requirements* (*specification*) *traceability* comprises those traces derived from or grounded in the requirements, and hence explicates the requirements' deployment process. *Pre-requirements* (*specification*) *traceability* comprises all those traces that show the derivation of the requirements from their sources, and hence explicates the requirements' production process. Only post-requirements traceability is evident in Fig. 8 since the requirement is the earliest development artifact available; this is the most common form of traceability in practice.

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*.

Figure 8 also serves to highlight some basic complexities surrounding traceability and so lends itself to the definition of a number of associated traceability concepts:

- Do we create an atomic trace for each class method or for the cluster of methods within a class? This is an issue of *trace granularity*.

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.*

- Do the three methods in the Display class fully satisfy the requirement? This is a question related to completeness. Does the trace then lead to the right code? This is a question of correctness. Is the trace up to date? This depends upon whether the traced artifacts reflect the latest project status. All of these questions are associated with the concept of *traceability quality*.

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.

- As Fig. 8 suggests, traces typically associate artifacts that are semantically very different, so the use of natural language alone to derive a trace link cannot always be trusted. For example, the play transition in the behavioural Statechart of Fig. 8 does not trace to the play method in the class diagram, or does it? Open issues in traceability research and practice have led to the formulation of a set of *traceability challenges* by the *traceability community*, and work is now underway to develop a *Traceability Body of Knowledge* (*TBOK*).

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

5 Conclusions

This chapter has defined terminology and concepts that are fundamental to the discipline of traceability. This includes the essential terms of trace, trace artifact, trace link, traceability and tracing in Section 2, along with a number of interrelated and dependent terms. The chapter has also described a generic traceability process model in Section 3 and characterised the basic activities involved in the life cycle of a trace. This includes a consideration of the activities comprising traceability strategy, traceability creation, traceability maintenance and traceability use. In Section 4, the chapter distinguishes between basic types of traceability and explains some key associated concepts.

The chapter is supplemented by an extensive glossary that has been developed and endorsed by members of the traceability community. This glossary contains additional terms and can be found as an appendix to this book.

References

Aizenbud-Reshef, N., Nolan, B.T., Rubin, J., Shaham-Gafni, Y.: Model traceability. IBM Syst. J. **45**(3), 515–526 (2006, July)

Boehm, B.W.: Software engineering. IEEE Trans. Comput. **c-25**(12), 1226–1241 (1976, December)

Cleland-Huang, J., Settimi, R., Romanova, E., Berenbach, B., Clark, S.: Best practices for automated traceability. IEEE Comput. **40**(6), 27–35 (2007, June)

Dorfman, M., Flynn, R.F.: ARTS – An automated requirements traceability system. J. Syst. Softw. **4**(1), 63–74 (1984, April)

Dorfman, M., Thayer, R.H.: Standards, Guidelines, and Examples on System and Software Requirements Engineering: IEEE Computer Society Press Tutorial. IEEE Computer Society Press, Los Alamitos, CA (1990)

Galvao, I., Goknil, A.: Survey of traceability approaches in model-driven engineering. In: Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference, Annapolis, MD, USA, 15–19 Oct, 2007, pp. 313–324.

Gotel, O., Finkelstein, A.: An analysis of the requirements traceability problem. In: Proceedings of the 1st IEEE International Conference on Requirements Engineering, Colorado Springs, CO, USA, 18–22 Apr, 1994, pp. 94–101.

Huffman Hayes, J., Dekhtyar, A., Sundaram, S.: Advancing candidate link generation for requirements tracing: The study of methods. IEEE Trans. Softw. Eng. **32**(1), pp. 4–19 (2006, January)

Lindvall, M., Sandahl, K.: Practical implications of traceability. Softw. Pract. Exp. **26**(10), 1161–1180 (1996, October)

Mäder, P., Gotel, O., Philippow, I.: Getting back to basics: Promoting the use of a traceability information model in practice. In: Proceedings of the 5th International Workshop on Traceability in Emerging Forms of Software Engineering, Vancouver, BC, Canada, 18 May, 2009a.

Mäder, P., Gotel, O., Philippow, I.: Motivation matters in the traceability trenches. In: Proceedings of 17th IEEE International Requirements Engineering Conference, Atlanta, GA, USA, 31 Aug–4 Sept, 2009b, pp. 143–148.

Naur, P., Randell, B. (eds.): Software engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7–11 October 1968, Brussels, Scientific Affairs Division, NATO (Published 1969)

The Oxford English Dictionary: Online Version, Oxford University Press, Oxford. http://www.oed.com. Accessed on January 2007

Pierce, R.: A requirements tracing tool. ACM SIGSOFT Softw. Eng. Notes. **3**(5), pp. 53–60 (1978, November)

Ramesh, B., Edwards, M.: Issues in the development of a requirements traceability model. In: Proceedings of the IEEE International Symposium on Requirements Engineering, San Diego, CA, USA, 4–6 Jan 1993, pp. 256–259.

Ramesh B., Jarke M.: Towards reference models for requirements traceability. IEEE Trans. Softw. Eng. **27**(1), 58–93 (2001, January)

Randell, B.: Towards a methodology of computing system design. In: Naur, P., Randell, B. (eds.) NATO Software Engineering Conference, 1968, Report on a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, pp. 204–208 (7–11 October 1968). Brussels, Scientific Affairs Division, NATO (Published 1969)

Winkler, S., von Pilgrim, J.: A survey of traceability in requirements engineering and model-driven development. Softw. Syst. Model. **9**(4), pp. 529–565 (2010, September). Springer (Published on line December 22, 2009)