Software Traceability: Trends and Future Directions

Jane Cleland-Huang *et al*

Published in Proceedings of the 36th International Conference on Software Engineering (ICSE), Hyderabad, India.

Cin/UFPE – Requirements Engineering
Danyllo Albuquerque and Sarah Moniky

Agenda

1. Introduction
2. Current Practice
3. Future Vision
4. Planning and Managing Traces
5. Creating and Maintaing Traces
6. Using Traces
7. Conclusion

1. Introduction

- Software traceability has long been recognized as an **important quality** of a well-engineered software system
- Defined by the Center of Excellence for Software and Systems Traceability (CoEST) as "*the ability to **interrelate** any uniquely identifiable software engineering artifact to any other, **maintain** required links over time, and **use** the resulting network to answer questions of both the software product and its development process*" [14]
- Traceability is a **required component** of the approval and certification process in most **safety-critical systems**.
  - **The DO-178C standard [73]** - which the USA Federal Aviation Administration (FAA) has established as the means of certifying airborne systems comply with airworthiness requirements.
  - **The USA Food and Drug Administration (FDA)** states that traceability analysis must be used to verify that the software design implements the specified software requirements.

1. Introduction

- In this paper, we set out a focused **research agenda for software traceability**.
- We based on previous work that first identified the **Grand Challenge of Traceability** [35] and then proposed a high-level roadmap in order to achieve it [34].
- Motivated by current practice, this paper draws out and drills down on the key areas in which research focus is needed.
- We provided details the state of the art in each of these areas and then focuses attention on specific high priority research needs

2. Current Practice

- To assess the current state of the practice, we reviewed **recent literature** describing industrial studies of traceability [63, 75, 71, 9].
- We also **asked** eight **seasoned practitioners** from the healthcare, enterprise information systems, military, and transportation sectors for their perspectives on **traceability practice and challenges**.
  - knowledge and acceptance of **traceability** tends to be **realized** on a **spectrum**, from fast-paced agile-like projects and/or business-oriented applications on one end, to slower-paced, carefully planned, safety-critical projects on the other.
  - The vast **majority** of practitioners do **not even understand** the "traceability" term. 5/8 IT project managers, had no understanding of the concept [8].

2. Current Practice

- There are several examples of traceability being achieved effectively:
  - **Panis and colleagues** listed the benefits as reduced effort during change management, coverage analysis (i.e. assessing missing requirements), and CMMI certification achievements. (Siemens TeamCenter)
  - **Nistala** described successful use of traceability across various phases of the life-cycle, and several of the practitioners we talked with described somewhat effective, albeit rather narrow, adoption of traceability to support specific tasks such as testing or regulatory compliance. (Tata Consulting Services')
- In short, **traceability** is successfully **implemented in some projects** within some organizations while the **majority of projects fail** to achieve effective traceability or incur excessive costs in so doing.

3. Future Vision

- CoEST's researchers/practitioners have worked since 2005 to **establish** for advancing the **state of practice** in software traceability [35, 34].
- This vision seeks to achieve **ubiquitous traceability** that is "always there" and "built into the engineering process."
- In this vision, the **cost** and effort of **establishing** and **maintaining** traceability basically **disappears** as trace links are generated automatically by tools as a by product of the development process.
- **Traceability information is made accessible** to humans to support the tasks that are relevant to their project environments, and rendered in ways that facilitate interaction and decision-making.
- **Ubiquitous traceability** is achieved automatically, as a result of collecting, analyzing, and processing every piece of evidence from which trace data can be inferred and managed.

3. Future Vision

- For example, a new developer joins an agile team and is assigned a user story to implement.
  - She uses automatically captured trace information to explore the impact of the new story on the system.
  - Results are quickly visualized in ways that help her to understand
    - (i) which parts of the codebase might need to be changed,
    - (ii) potential effects on existing user stories and test cases, and
    - (iii) a list of team members who have previously worked with the code and could be considered expert consultants
- We structure our presentation of future research needs around the three perspectives of ***goals, process, and technical infrastructure***, each of which has a unique impact upon traceability.

3.1 A Goal-Oriented Perspective

- An earlier roadmap [34], produced by researchers in the traceability community, identified a number of challenges for traceability, including the *Grand Challenge to achieve Ubiquitous Traceability*.
- These challenges are reproduced as goals in **Table 1**.
- Each **goal** represents a **desired quality** of traceability, and is refined into a set of research topics and practices (described in detail in [35]).

3.1 A Goal-Oriented Perspective

Table 1: A Goal-Oriented Perspective

Goal 1: Purposed	**Traceability is fit-for-purpose and supports stakeholder needs (i.e., traceability is requirements-driven).** Prototypical stakeholder requirements for traceability need to be clearly defined and measurable for specific software and systems engineering tasks.
Goal 2: Cost-effective	**The return on investment (ROI) from using traceability is adequate in relation to the outlay of establishing it.** Develop techniques for computing the ROI of traceability in a project, understanding the impact of various traceability decisions at various stages of the life-cycle upon both the cost and benefits of the traceability process.
Goal 3: Configur-able	**Traceability is established as specified, moment-to-moment, and accommodates changing stakeholder needs.** Develop techniques for dynamically generating and maintaining accurate and semantically rich trace links that are configured according to the current needs of the project.
Goal 4: Trusted	**All stakeholders have full confidence in the traceability, as it is created and maintained in the face of inconsistency, omissions and change; all stakeholders can and do depend upon the traceability provided.** Develop techniques for assessing and communicating the current state of traceability in a project, and develop self-adapting techniques so that quality is preserved in the face of change.
Goal 5: Scalable	**Varying types of artifact can be traced, at variable levels of granularity and in quantity, as the traceability extends through-life and across organizational and business boundaries.** Develop techniques for scaling up traceability techniques, and for supporting multi-grained traceability across a variety of artifact types and organizational boundaries.
Goal 6: Portable	**Traceability is exchanged, merged and reused across projects, organizations, domains, product lines and supporting tools.** Develop policies, standards, and formats for exchanging and integrating traceability information across projects and organizations.
Goal 7: Valued	**Traceability is a strategic priority valued by all; every stakeholder has a role to play and actively discharges his or her responsibilities.** Develop supporting techniques that cross the technical and business domains of a project so that the benefits of traceability are visible and accessible to all stakeholders.
Grand Chal-lenge: Ubiquitous	**Traceability is always there, without ever having to think about getting it there, as it is built into the engineering process; traceability has effectively "disappeared without a trace."** Achieved only when traceability is established and sustained with near zero effort.

3.2 A Process-Oriented Perspective

- The **process-oriented perspective** (Figure 1) focuses upon the primary areas associated with the traceability life-cycle:
  - Planning and managing a traceability strategy, and
  - creating, maintaining, and using traceability [36].
- At the start of a project, the traceability strategy is **planned**, and the project environment is instrumented accordingly.
- The project's traceability is then **implemented** and **managed** as the project proceeds. Trace links are created, used, and maintained according to the strategic plan.

3.2 A Process-Oriented Perspective

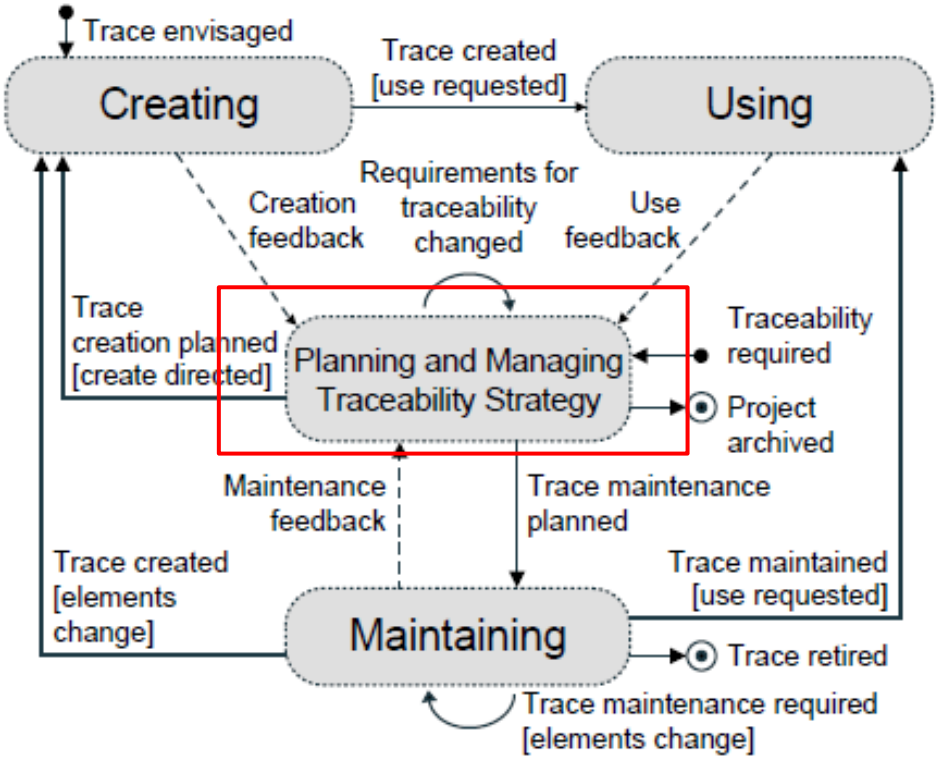


Figure 1: The Process Perspective

3.3 A technical-Infraestructure Perspective

- The significant practical challenges associated with **instrumenting** a **project environment** for traceability are highlighted by this perspective.
- A tracing context must minimally include functions for storing and retrieving physical data; supporting strategic planning; physically creating and maintaining trace links; executing trace queries; and interacting with the trace user. (Figure 2).
- Infrastructure and supporting algorithms must therefore be developed to support cross-organizational traceability and data integration across a wide variety of tools and data formats.
- If such issues are not addressed, it is unlikely that advances in traceability research will be able to make a significant impact on industrial practice.

3.3 A technical-Infraestructure Perspective

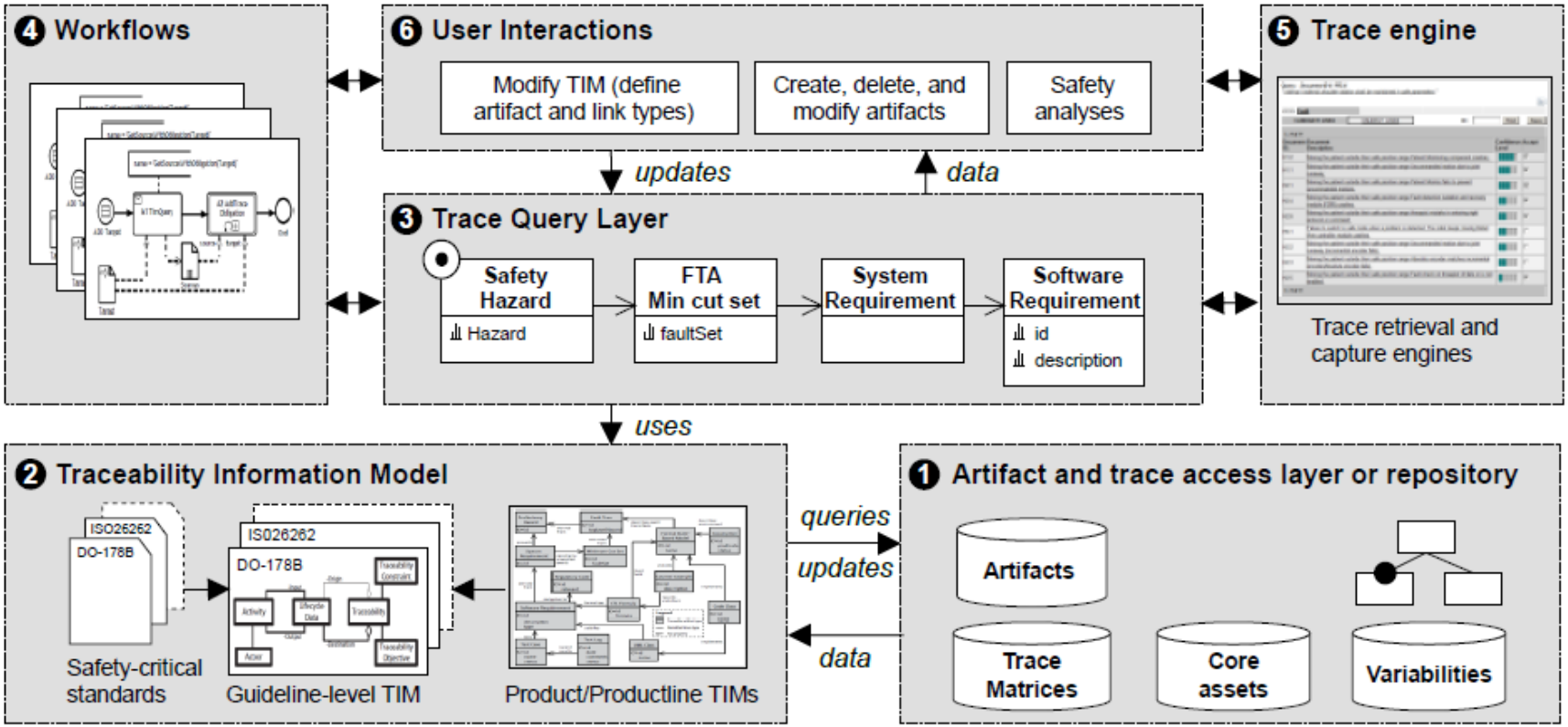


Figure 2: The Technical Perspective

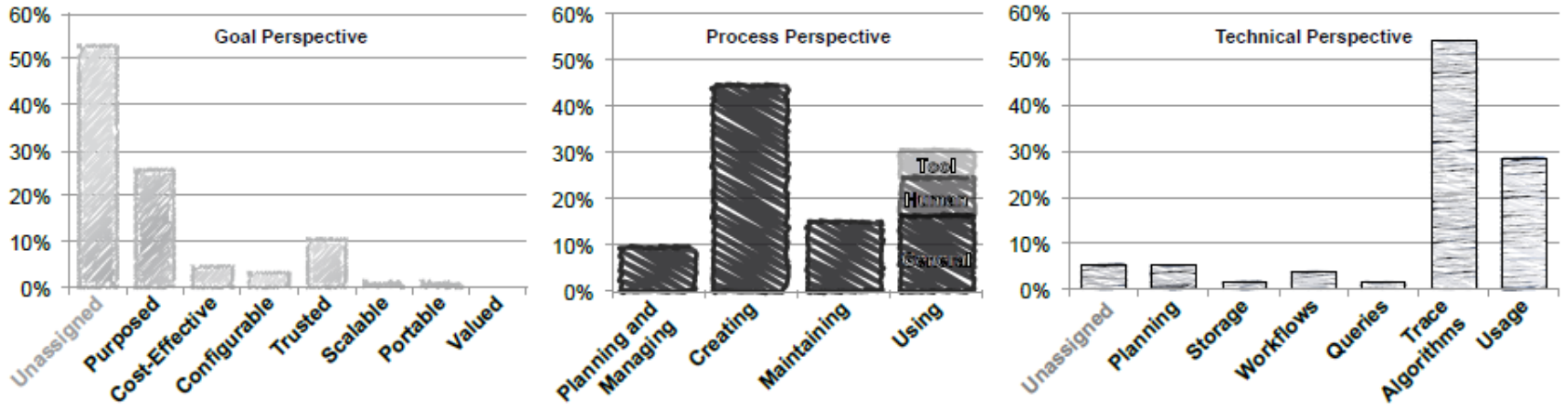
3.4 Integrating Perspectives

- Take the **scalable goal** as an example
- ***At the goal level*** **-** A scalable tracing solution seeks levels of abstraction/granularity in traceability techniques, facilitated by improved trace visualizations, to handle large datasets and its longevity [34].
- ***At the process level***, this means solutions that help stakeholders to determine the right level of granularity and to facilitate tracing at that level.
- ***At the process level*****,** the infrastructure needed to support this functionality needs to be embedded across multiple tools and environments. An emphasis on how the stakeholder interacts with and handles issues of scale is a given.

3.4 Integrating Perspectives

- **DBLP search** of selected publications from 2003-2013 (IEEE Transactions on Software Engineering, ASE, ICSE, RE, ASE, Software, and ICSM) was carried out using the search **term Trace**.
- Filtering out papers not directly related to software traceability returned 72 papers.
- Each paper was then classified according to elements of the goal, process, and technical perspectives (results in Figure 3).

3.4 Integrating Perspectives



(a) Goal perspective (b) Process perspective (c) Technical perspective

Figure 3: Traceability Research by Perspective over the past decade (2003-2013)

3.4 Integrating Perspectives

- It would be a mistake to correlate **research effort with criticality** (or priority) of the addressed problem.
- Researchers may **choose** to focus on **specific problems** for a number of reasons including
  - (1) importance of the problem,
  - (2) availability of data,
  - (3) interests of the research group, and
  - (4) perceived ability to publish results.
- Factors 2-4 all serve to inhibit researchers from addressing certain types of pressing research challenges.
  - For example, researchers have demonstrated a propensity to favor the **trace creation process** area as datasets are readily available, and research questions can be easily formulated and experimentally evaluated.
  - In contrast, other equally critical research problems, such as those related to **processes for traceability strategizing**, tend to attract less research attention

3.5 Research Focus Area

- Based on these findings, we set out to identify compelling areas of traceability need. We have identified 7 specific areas for research.
- We describe these in the context of the **process-oriented perspective** and make connections to the other **two perspectives**. We focus upon
  - Section 4 – Planning and Managing
    - (1) understanding stakeholder needs,
    - (2) developing techniques to support traceability strategizing,
  - Section 5 – Creating and maintaining traces
    - (3) trace creation,
    - (4) trace maintenance,
    - (5) trace integrity,
  - Section 6 – Using traces
    - (6) accessing and querying trace data,
    - (7) visualizing trace data.

3.5 Research Focus Area

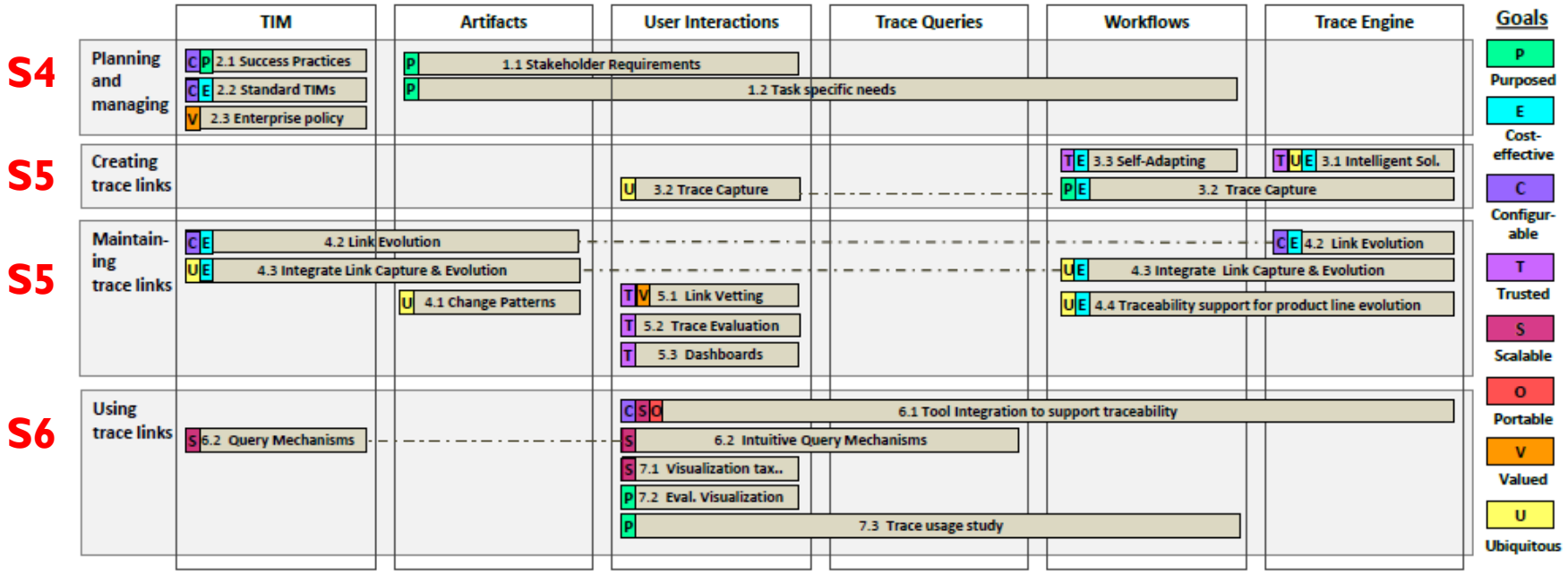


Figure 4: Research Directions mapped to the Three Perspectives

Figure 4 maps each of the research directions onto their relevant process and technical areas. It also highlights the quality goals they address.
For some of these goals we provide explicit mappings throughout the remainder of the paper, while others represent more general relationships and are shown only in the diagram

4. Planning and Managing

- There is no "**one-size-fits-all**" approach to software traceability.
- Projects of different types, sizes, and criticality levels all have their own reasons for requiring traceability and their own unique constraints that influence its implementation.
- Traceability **planning and managing** involves
  - determining the *stakeholder* and system *requirements* for traceability on a project,
  - designing a suitable *traceability strategy* to enable them to be satisfied [36].
- These are the first two focal areas in which research is needed.

4.1 Understanding Stakeholder Needs

- Traceability research must be driven by **the needs of its stakeholders**, who ultimately adopt tracing solutions to support activities such as requirements satisfaction assessment, impact analysis, compliance verification, or testing effort estimation.
- These tasks are performed by a **variety of stakeholders** including requirements analysts, safety analysts, certifiers, reverse engineers, developers, architects, maintainers, and Verification and Validation (V&V) analysts (possibly Independent).
- Unfortunately, there is little prior work that examines the **specific needs of the stakeholder in the traceability process** and, as a result, academic researchers are left making assumptions about industry needs

4.1.2 Task-Driven Traceability

- Unfortunately, the traceability community has not invested much effort to understand tracing tasks at detailed level.
- **Ramesh and Jarke** [78] conducted an industrial study and developed several traceability reference models that captured artifact types and the links needed to support specific tasks (e.g. requirements to design)
- Other researchers have investigated the use of traceability within specific contexts.
  - **Von Knethen** [89] and Conte de Leon [18] both described techniques for using trace links to support impact analysis
  - **Mader et al**. [60] and **Poshyvanyk et al.** [77] explored its use for supporting software maintenance and for bug-fixing [60].

4.1.3 Research Directions

- This area of research is driven by the goal for traceability to be ***purposed***. From the process perspective, ***planning and managing*** is at the heart of and influences all other areas of the traceability life-cycle. It impacts the ***user interactions*** layer of the technical infrastructure perspective.

- Main Research Directions (RD):
  - **RD-1.1 Develop prototypical stakeholder requirements** for traceability, including scenarios of use.
  - **RD-1.2 Empirically validate task-specific traceability techniques** as applied by stakeholders.

4.2 Traceability Strategizing

- **Without an upfront tracing strategy**, projects tend to produce *ad-hoc*, inconsistent, incomplete, conflicting trace links even in many safety-critical projects [64].
- Given the cost-benefit debate surrounding traceability [5, 46], the natural tendency is to put only absolute essential traceability in place to address the immediate concerns and visible needs of a project.
- For example, traceability may be established immediately prior to the external certification or approval process in a safety-critical project, instead of systematically throughout the entire development process.

4.2.1 Traceability Guidelines

- While bodies of knowledge, handbooks, and process improvement frameworks describe the need for **traceability in general terms**
  - (e.g. BABOK, CMMI, INCOSE Handbook, IREB, PMBOK, SPICE, SWEBOK, etc.),
- While international standards routinely **demand** traceability
  - (e.g. IEEE-STD-830-1998, IEEE-STD-1220-2005, IEEE-STD-15288-2008, ISO-29148-2011)
- **Explicit guidance and assessment** on particular practices for setting up traceability is **scarce**.
- The landscape is better in the safety-critical domain, where standards such as the FAA's DO-178c [29] provide detailed traceability guidelines for various levels of system criticality.

4.2.2 Reference Models

- The most significant contribution of the research community to traceability strategizing has been on reference traceability for standard projects.
- A traceability reference model specifies
  - the permissible artifact types and permissible link that can form a trace on a project,
  - and is derived from an analysis of the queries that the resulting traceability is intended to answer [78].
- Despite several proposed traceability reference models, there is none that is universally accepted or widely used in industry, neither generic nor domain-specific.
- An exception to this could become the work of Katta [48], which denes a detailed reference model for use in the highly-regulated nuclear domain.

4.2.3 Traceability Information Models

- The resulting **Traceability Information Model** (TIM) is effectively an abstract expression of the intended traceability for a project [36].
- Work on how TIMs can be designed, adapted and employed as part of an end-to-end traceability process is now emerging.
- Case study reports provide mature insights into industrial practices for planning and managing traceability [51, 79, 78, 75, 71, 38], suggesting that the potential for researchers and practitioners to work together to synthesize lessons and inform practice is now mature.

4.2.4 Strategy Customization

- Because **systems evolve over time -** and because perfect traceability cannot be planned for upfront **-** self-managing traceability systems will need to **configure, grow and repair** organically to address changing contexts and needs.
- Traceability strategizing will be intrinsic to project management practices, facilitated by intelligent toolkits that help to devise just the right traceability strategy based on an assessment of evolving needs and available resources.
- Strategies will be designed dynamically to meet the traceability needs and economic pressures of any particular project's context.

4.2.5 Research Directions

- Traceability strategizing is primarily related to the ***Cost-effective*** goal which states that the "The ROI from using traceability".
- It anchors ***planning and management*** processes and is supported primarily in the ***Traceability Information Model*** layer of the technical infrastructure.
- We define important research in the area as follows:
  - **RD-2.1 Identify ingredients for traceability success** in different contexts, from an understanding of industry best and worst practice, and then use this knowledge to establish a process framework to guide practitioners, develop standards, inform tools, and enable training.
  - **RD-2.2 Prepare a family of standardized TIMs** and usage guidance. Adaptable and extensible meta-models need to provide the capability for a project or organization to grow its traceability competence via well-defined paths.
  - **RD-2.3 Develop policies and protocols** that enable the traceability of any desired corporate asset to be planned for and established across the enterprise..

5. Creating and Maintaing Traces

- As previously discussed, over **50%** of the traceability related research papers have focused on the creation and/or maintenance of trace links.
- We describe the **three essential research** areas of
  - (3) trace creation,
  - (4) trace maintenance, and
  - (5) trace integrity
- All of which must work synergistically in order to **automate the trace creation process** and work toward the goal of **completely eradicating manual traceability effort**

5.1 Trace creation

- Current work in the area of trace creation primarily falls under the two distinct categories:
  - (1) trace retrieval, and
  - (2) prospective trace capture.
- We discuss each of these

5.1.1 Trace retrieval

- The idea behind trace retrieval is to dynamically generate trace links between source and target artifacts. Exemplifying:
  - (i) retrieve all relevant Java classes for a given requirement,
  - (ii) retrieve all requirements that demonstrate compliance to a specific regulatory code.
- The current approaches are based on seminal work of Antoniol *et al.* [3], who used a probabilistic approach to retrieve trace links between code and documentation.
- The problem is primarily caused by term mismatches across documents to be traced. Looking ahead, we therefore need to explore alternate trace creation techniques that circumvent limitations of term mismatches.
- Three promising approaches include incorporation of (1) runtime trace information, (2) general and domain specific ontologies, and (3) special case strategies.

5.1.2 Prospective Trace Capture

- **Another** orthogonal **approach** to trace creation **sets out to capture and infer trace links** from the project environment and from the actions of the project stakeholders.
- This approach is appealing because trace links can be inferred as a natural byproduct of software engineering activities.
- Work in this area is divided between
  - techniques which instrument the general project environment to monitor the actions of developers [6] and
  - those which infer trace links from tagged items in the logs of version control systems and other kinds of repositories [40, 23].

5.1.3 Self Adaptation

- Self-aware systems are able to modify their own behavior in an attempt to optimize performance. Such systems can self-diagnose, self-repair, adapt, add or remove software components dynamically, etc. [88].
- Initial work has investigated adaptation in traceability environments.
  - **Poshyvanyk et al.** used a Genetic Algorithm (GA) to discover the best way to parameterize Latent Semantic Indexing (LSI) [74].
  - **Falessi et al.** [28] used regression analysis to discover the right combination of techniques for a given dataset.
  - **Lohar et al.** [58] modeled available features of the trace engine in a feature model, and used a genetic algorithm to search for the ideal configuration.

5.1.4 Research Directions

- To work toward achieving automated trace creation, we propose the following priority research.
  - **RD-3.1 Develop intelligent tracing solutions** which are not constrained by the terms in source and target artifacts, but which understand domain-specific concepts, and can reason intelligently about relationships between artifacts.
  - **RD-3.2 Deliver prospective trace capture solutions** that are capable of monitoring development environments, including artifacts and human activities, to infer trace links.
  - **RD-3.3 Adopt self-adapting solutions** which are aware of the current project state and reconfigure accordingly in order to optimize the quality of trace links.

5.2 Trace maintenance

- One of the greatest challenges of traceability in practice is that of maintaining trace links as a system evolves.
- Trace maintenance is essential regardless of whether trace links have been created manually or with tool support

5.2.1 Evolving Links

- The challenge is to evolve trace links automatically as related artifacts change.
- As shown in our literature study, this area of research has garnered less than one third of the effort ascribed to trace creation problems.
- Prior work has focused on (I) heuristic and (II) trace retrieval methods.
  - (i) Mader et al. explored the use of heuristic techniques that recognize specific development activities, such as changes applied to a UML model [61], and then evolve trace links according to a set of heuristics.
  - (i) Similar approaches have been explored for changes in requirements [11], and between architectures defined using xADL and source code [70].
  - (ii) Researchers have also developed trace retrieval approaches which attempt to identify deltas between the artifacts retrieved over consecutive traces [92]

5.2.2 Research Directions

- The following research directions address the challenges of link evolution.
  - **RD-4.1 Understand patterns of change** across various artifacts including requirements, design, and code.
  - **RD-4.2 Develop heuristics and probabilistic approaches** for evolving trace links as artifacts change.
  - **RD-4.3 Integrate prospective capture with link evolution** techniques.
  - **RD-4.4 Develop traceability structures to support the evolution of products across a product line**.

5.3 Trace Integrity

- Trace integrity is concerned with validating the correctness of trace links that have been created and maintained, and/or communicating the quality of an existing set of links.
- There are three primary techniques related to trace validation at present
  - (1) eliciting feedback from human analysts,
  - (2) exploiting the semantics and context of each trace link, and
  - (3) computing metrics which serve as indicators of quality.
- These areas provides support for improving and understanding the integrity of trace links, but each also presents its own research challenges

5.3.1 Improving Integrity through Human Feedback

- For many tasks, analysts need to evaluate the generated trace links.
- Several studies have shown that human feedback is incorrect approximately 25% of the time, which can negatively impact the quality of the generated trace links [53].
- Studies have also shown that humans use different strategies when examining trace links, such as accepting the first good link they find without looking further in the list.
- Some strategies require high effort while resulting in low accuracy [53].
- A limited number of state-of-the-art tracing tools integrate user feedback. The ADAMS [20], POIROT [57], and RETRO [44] research tools collect relevance feedback on trace links that have been generated automatically using information retrieval techniques

5.3.2 Toward Automating Link Evaluation

- Given the potentially large number of trace links in a project, and the cost and effort of validation, it is appealing to consider techniques for assigning confidence scores to each link or to a specified set of links.
- We see three main areas of work in this area.
  - (i) The semantics of a trace can be analyzed to understand the rationale of the link
  - (ii) Artifacts of the same type can be analyzed in order to draw conclusions about the consistency and conclusiveness of existing and missing traces
  - (iii) We need to develop an understanding of how much trust is required based on the intended use of a trace so that the quality of the link can be evaluated within a meaningful context.

5.3.3 Determining and Comunicating Confidence

- it is pragmatic to recognize that as we cannot guarantee complete and accurate traceability, we should devise techniques for clearly communicating confidence levels to the stakeholders.
- Another interesting research area therefore involves the creation of metrics to evaluate the overall quality of a collection of trace links.
- Recent work by Rempel *et al.* generated a TIM from traceability data in a number of safety-critical projects and used formal logic to compare it to the TIM prescribed by relevant process guidelines.

5.3.4 Research Directions

- The following research would serve to improve a stakeholders' ability to assess trace integrity.
  - **RD-5.1 Develop human-centric tools to support link verification**.
  - **RD-5.2 Develop algorithms and supporting tools for automatically evaluating** the correctness of existing trace links, whether created manually or with tool-support.
  - **RD-5.3 Create visual dashboards** to visualize the traceability quality of a project.

6. Using Traces

- Trace links are created to empower trace users to perform various software engineering tasks more effectively.
- We explore research that focuses on enabling stakeholders to
  - (6) Access and query trace data,
  - (7) Presenting trace results for decision-making purposes, through targeted visualizations

6.1 Acessing and Query Trace Data

- Requirements management tools often provide support for tracing requirements to other artifacts in the software development life-cycle.
  - For example, IBM's RequisitePro allows developers to relate requirements kept within the tool to other tools within the product suite, such as Rational Software Modeler.
  - An all-lifecycle-management (ALM) tool or platform would be an ideal project backbone for through life traceability, from an academic perspective, as a single repository would keep all the artifact types and link data.
- An industry study showed that many organizations and users prefer chains of task-specific tools to suit their development preferences [9].
- Projects therefore tend to adopt a wide variety of CASE tools across different organizations.

6.1.1 Support for Heterogenity

- The ability to access and query heterogeneous trace data is a cross-cutting and enabling requirement for applying end-to-end traceability in real world development contexts.
- It forms a basis for all seven goals that support the Traceability Grand Challenge (see Table 1).
- It is also a prerequisite for making proper use of traceability from a process perspective (see Figure 1)

6.1.2 Creating and Reusing Trace Queries

- One of the greatest inhibitors of trace usage is the challenge of formulating complex trace queries.
- Many useful trace queries are quite complex and are not supported by requirements management tools.
- As a result, users are often required to formulate complicated queries using SQL or other similar languages.
- To address these problems, several researchers have explored alternate query languages and query reuse mechanisms. See works on this section.

6.1.3 Research Directions

- Research here is needed in a couple of directions.
  - **RD-6.1 Integrate existing development tools** and other relevant data that is created as part of a development project.
  - **RD-6.2 Provide intuitive forms of query mechanism** that do not require specialized training. These may include simpler techniques for formulating new queries or retrieving existing, reusable, trace queries

6.2 Vizualizing Trace Data

- Traceability is put in place to establish useful links between the artifacts of a development process, so that the eventual traversal of these links can support various engineering tasks.
- At present, trace information can be difficult for developers to use, since little research attention has been directed toward its presentation and eventual end-use.
- The most common way to enter and show traceability information in practice remains the trace matrix [56], a two-dimensional view on to a multidimensional information space, despite the well-known difficulties associated with its scalability.

6.2.1 Trace Visualization

- Enormous advances have been made in popular techniques and tools for information and knowledge visualization [42].
- Visual analytics are now a common form of support for decision-making activities in many fields of endeavor [87] and advice on selecting suitable visualization techniques is readily available [50, 82, 76].
- More generally, linear, tabular, hierarchical, and graph-based representations are the most prevalent forms of visualization used to depict traceability information in commercial tool support, and hyper-links (cross-references) are routinely used to associate artifacts and traverse the links interactively.

6.2.2 Selecting Trace Visualizations

- Concurrent with the development and evolution of techniques, traceability researchers have emphasized the need to better understand the users, tasks, and project constraints that drive the development and selection of suitable visualizations for traceability purposes [66, 39, 91, 16].
- A recent empirical study examined four common visualizations used to present traceability information (matrices, graphs, lists, and hyperlinks), to investigate which ones were better suited to which tasks [56].
- It found that matrices and graphs were preferred to support management tasks, while hyperlinks were preferred to support implementation and testing tasks.
- Such studies bring an important focus to task-driven visualization, the longer-term objective of this research being to suggest the most appropriate traceability visualization(s) for any task at hand.

6.2.3 Research Directions

- We envisage a future in which there are better visual ways to interactively define, create, maintain, analyze, and use traceability information effectively.
- Necessary research comprises three directions.
  - **RD-7.1 Construct a taxonomy of available visualizations and fundamental traceability tasks.**
  - **RD-7.2 Gather and share user-based empirical data** to evaluate trace visualizations and direct the formulation of novel ones.
  - **RD-7.3 Perform in-situ user studies to evaluate and understand task-specic needs for trace information** and develop novel ways to provide the needed information to stakeholders.

7. Conclusion

- We have identified seven **research areas** and their associated "**research directions**" which must be addressed in order to achieve the grand challenge of **ubiquitous traceability**.
- Each of these **research directions** is fully actionable and will help our community to work collaboratively towards advancing the state-of-the-art and state of practice in traceability.
- The identified research directions are quite varied in nature. Some focus on algorithmic solutions, others on process improvement, and still others on technical infrastructure needs.
- Advancing the state-of-the-art in software traceability will require the cooperation of researchers with different skill-sets from areas as diverse as information systems, data mining, visualization, and systems engineering.