From Farm to Fork or a Bite of the Unknown: Learning from the Food Industry

O.C.Z. Gotel
Department of Computer Science
Pace University
New York
ogotel@pace.edu

S.J. Morris
Department of Computing
City University
London
sjm@soi.city.ac.uk

ABSTRACT

Establishing requirements traceability within a software development project is a problem that has been explored, discussed and addressed by both academics and practitioners for many years. However, progress has been fragmentary and slow to date, an observation that has been reflected in the release of a recent document that attempts to articulate what the community considers to be the "Grand Challenges in Traceability". In other domains, such as the food industry, traceability is frequently a regulatory demand and routinely achieved for many of the products that we eat. This paper examines the meaning and provision of traceability in the food industry, at an initial and high level, as a point of comparison through which to motivate possible explanations for the unique difficulties of establishing traceability in software engineering. The goal of this paper is to encourage a closer investigation into traceability concepts and practices from other domains, and to thereby trigger a wider discussion about the insights and possible lessons for the software industry.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements / Specifications.

General Terms

Management, Documentation, Theory.

Keywords

Food Traceability, Requirements Traceability, Trace, Traceability, Traced Object, Trace Relation.

1. INTRODUCTION

A cursory examination of the software engineering literature over the past fifteen years, and more particularly of the requirements engineering literature, reveals that traceability is a focal area of interest and concern for many people, resulting in a series of workshops on the topic [20, 21, 22]. Papers over this period have ranged from examining the needs for and the problems associated with traceability [1, 11], to practical experiences with establishing traceability in the field [2], to the use of information retrieval based techniques for recovering traceability linkages after the fact [13]. Despite significant advances in many important areas, this community of academics and practitioners has recognized that efforts have not yet resulted in the progress that is desired [4].

'Traceability' is a term that has applicability in domains beyond software engineering. For example within metrology, the field of knowledge concerned with measurement, establishing traceability necessitates making a documented case to demonstrate that a new measurement result relates back to some agreed national or international standard unit of measurement "through an unbroken chain of calibrations of a measuring system or comparisons, each contributing to the stated measurement uncertainty" [14]. Within the food industry, the need to know what is referred to as the *backstory* of a food product is nowadays mandated to satisfy legal regulations for food safety [5]. Popular taglines, such as "from farm to fork", permeate the media and reveal the necessity to render visible the entire supply chain of a food product, from its raw source ingredients, through production, processing and distribution [6]. Coupled with the increasing demand to either assure or certify traceability in domains like these is the apparent ability to achieve this quite satisfactorily in practice [10].

This paper examines traceability in the context of the food industry and contrasts this with that of the software industry to help suggest where some of the challenges may lie. It considers the motivation for providing traceability, the processes typically undertaken to accomplish it, the concepts used to characterize its provision, the objects of traceability interest, the underlying meaning of a trace and the nature of trace relations. The paper suggests that in vernacular use there is a clear distinction between literal and figurative definitions of 'trace', and this is a key difference that has implications for how we need to think about traceability and about what is achievable in these two industries.

2. WHY TRACE?

In the food industry, traceability is predominantly driven by the demand for food safety. *Traceback* and *traceforward* can facilitate the recall and withdrawal of food, helping to guarantee the sources of a food product or the quality of a processing step, thereby eliminating any hazardous pathways [7]. The value of traceability in the food industry therefore increases when the

TEFSE/GCT'07, March 22-23, 2007, Lexington, KY, USA.
Copyright 2007 ACM ISBN 1-59593-6017/03/07...$5.00.

health risk associated with certain foods increases and where penalties can be incurred, though there will always be occasions and places when consumers are ready to taste the unknown or to simply take risks. A second driver is the fact that there are specific claims about food that cannot be seen by simple inspection, like organic sourcing and fair trade practices. These need to be demonstrated via a documented record and are usually certified by an external third party to promote consumer trust, as per the International Federation of Organic Agricultures Movements or the Soil Association Organic Standards. Traceability in the food industry is therefore also required to validate the presence or absence of attributes about food, to differentiate products for marketing purposes and to provide for customer reassurance. The importance of traceability here is linked to the potential market size for products with these valued attributes, along with the premium that customers would be prepared to pay for them.

In theory, traceability is expected to assist with a similar set of goals in the software industry (and is recommended as a general principle of software validation by the same US agency requiring food traceability [8]). It enables the determination of what parts of a software product are impacted by a change during development and maintenance, the demonstration of compliance to processes and standards, and the verification that specified requirements and properties are present in the evolving software product. It can therefore help manage risk and demonstrate the presence of properties too. However, providing traceability to help recall a food product, or to pinpoint the source of a food scare so as to replace an ingredient, is possibly a less complex task than using it to make on-the-fly changes to evolve an *in situ* product. Part of the difficulty of establishing traceability in the software industry is the complexity of the core activity that traceability is expected to assist with, namely managing the multi-various changes inherent in such processes. Also, the attributes of a food product are somewhat inherent, in that an apple is either organic or not (if the record shows so). Those properties pertaining to software, like reliability, also need to be demonstrated, but just because they were demonstrated at one point in time does not guarantee they still hold at another point in time or within a wider compositional context. Are similar issues not also apparent in the food industry?

3. TRACEABILITY PROCESSES

There are many parties, with well-defined roles, involved in the whole lifecycle of any food product (e.g. sources, processors, recipients, transporters, etc.) While there is often no end-to-end responsibility for traceability by one party, there are some simple and generic traceability principles that all parties need to comply with, an example of which is specified in the Food Safety Regulation EC178/2002 [5]. Paraphrasing this, all parties subscribe to 'one up / one down' traceability [3]. This is the agreement to maintain records about the immediate supplier and the immediate subsequent recipient of a food product at all points along the supply and distribution chain, but not with legal dictate as to which method they are to use. These *external* traceability links record the input and output (i.e. the immediate origin and immediate destination information) of a product, like the raw materials used by suppliers. *Internal* traceability is a concept restricted to the transformations that take place under the control of the party concerned, though not every party materially alters the food product while it is under their jurisdiction. The responsibility for through-life traceability is thus distributed and the health of the chain is a jointly shared achievement. When many individual food products are supplied via the same chain, there are also economies of scale involved and supply management quality incentives. A weak link, in the form of an uncooperative, non-compliant or non-trusted party, means the whole chain suffers.

Within the software industry, the responsibility for establishing traceability is not usually distributed and shared between all participating parties, but tends to be the responsibility of a few (if any). Short of following a rigid waterfall process model, the delineation of the input and output to developmental activities is not as clear cut as in the food lifecycle, so responsibility can become blurred when developing a software product with on-going feedback and iteration. A lack of simple protocols and agreements further makes it difficult to trace across functional disciplines and organizational boundaries. In addition, with an individual software product, there are rarely the same economies of scale and shared risks to reputation or finances motivating the perfection of the chain, unless software process improvement and associated certification is a primary goal of all parties. The question of who does the traceability and who benefits has long been asked, along with debate about return on investment, leading to research on the automated recovery of traces. The alternative is, theoretically of course, the automatic generation of traces as a by-product of software development processes. Could the distributed management of traces, according to a simple process as per the food industry, become a viable proposition in the future?

4. TRACEABILITY CONCEPTS

While the basic principle of 'one up / one down' traceability applies in the food industry, there are factors that influence the amount of work that needs to be done on a case-by-case basis [9]:

Depth – How far back or how far forward to trace in the lifecycle of food. As the depth increases, traceability becomes more difficult and uncertainty grows, especially with the more complex transformations a product undergoes.

Breadth – The amount of information recorded about a product (i.e. the number of attributes or amount of meta-data). As the breadth increases, the harder it becomes to manage traceability.

Precision – The level of detail (i.e. granularity) recorded about individual attributes. The greater the precision, the higher the overhead and the more difficult the traceability is to manage.

The cost of doing traceability in the food industry is clearly correlated with decisions about the required interplay between depth, breadth and precision. Finding the right balance for these three dimensions for any one product depends on the value that traceability is seen to add and this will vary according to risk and attribute premium, as explained in Section 2. For example, is it cost-effective to maintain the ability to trace the pieces of an apple in an apple pie back to the source tree or would back to the orchard, grower or region suffice? Is it commensurate with the risk to shut down an entire industry if one product or processing step is found wanting? If the concern is to certify the apple as organic, does information need to be maintained about orchard fertilizers and the 'picked-from-the-tree' date? The costs of traceability cannot be divorced from the record-keeping tasks it relies upon and the benefits cannot be ascertained without clearly stated goals.

In the software industry, there have been a few attempts to clarify the dimensions of requirements engineering [18]. Traceability is also required to help achieve specific goals and can obviously be achieved at many levels. A similar simple language may be useful for articulating the required traceability dimensions *a priori*:

Depth – Is it necessary to trace back to and from source documents or is the need simply to link requirements to design?

Breadth – Does associated rationale or costing information need to be traceable for all or for some subset of the requirements?

Precision – If so, should the required rationale information take the form of a one-line explanatory sentence or should it be an intricate form of structured argumentation?

5. TRACED OBJECTS

This paper differentiates between three terms in an attempt to clarify the objects of traceability interest within the two domains:

i) The 'traced (or 'to-be-traced') object', the material object or abstract entity whose path has been, or must be, traced or tracked (e.g. the animal in the wild moving across terrain or a specific functional requirement). This is the primary object of traceability interest and no assumption can be made *a priori* that the object is in fact traceable.

ii) The 'trace', the marks left behind, if any (e.g. the track on soft ground or apparent implementation of a requirement).

iii) The 'trace', or more specifically the 'trace record', somehow generated specifically for its own sake (e.g. a plaster cast of a paw print or the documentary record of the transformation of a requirement into a specific downstream component).

A traced (or to-be-traced) object is called a 'lot' in the food industry. A lot can refer to either an individual piece or a batch (i.e. a collection or composite of pieces). They are tangible and visible materials, namely food stuffs. Attached to the physical object is information about both it and the processes involved in its creation, processing and distribution. It is obviously much easier to attach this meta-data directly to the physical object if it is in a form that can be directly labeled (e.g. an apple with a grower's sticker) or if it is packaged (e.g. a boxed apple pie with a bar code). In this way, there is no physical separation between the object and the meta-data maintained about it. Together they form the more encompassing trace record which is also to-be-traced.

The unusual, and most challenging aspect of traceability of any component that forms part of a software development process, is the indirect, non-physical and intangible nature of many to-be-traced objects. This problem is implicit in the fact that the process always involves representations of some form rather than physical objects, and these representations themselves seldom share a common origin or form. This distinction has implications that are explored in later sections and is illustrated in Figure 1 below.

6. WHAT EXACTLY IS A TRACE?

The online version of the Oxford English Dictionary shows at least three dozen different definitions for the term 'trace' [16]. The most common literal definitions of 'trace' include:

a) *"Vestiges or marks remaining and indicating the former presence, existence, or action of something."*

b) *"An indication of the presence of a minute amount of some constituent in a compound; a quantity so minute as to be inferred but not actually measured."*

c) *"The detailed examination of the execution of a program or part of one with the aid of another program that can cause individual instructions, operands, and results to be printed or displayed as they are reached by the first program*."

What is common to these definitions is that such 'traces' exist by virtue of direct evidence or documentation: a) a paw print; b) wetness on the ground after rain; or c) a series of recorded values.

The food industry is interested in a chain in which a to-be-traced [(i)] object is identified as a component at a particular stage and then as a 'trace' [(ii) or (b)] in the next stage because it survives in a new form as a result of some type of transformation (or perhaps in a completely unaltered form but altered in external appearance by a wrapper, or simply by location when shipped from a factory to a warehouse). That 'trace' [(ii)] then itself becomes a new to-be-traced [(i)] object, and so on. This chain-

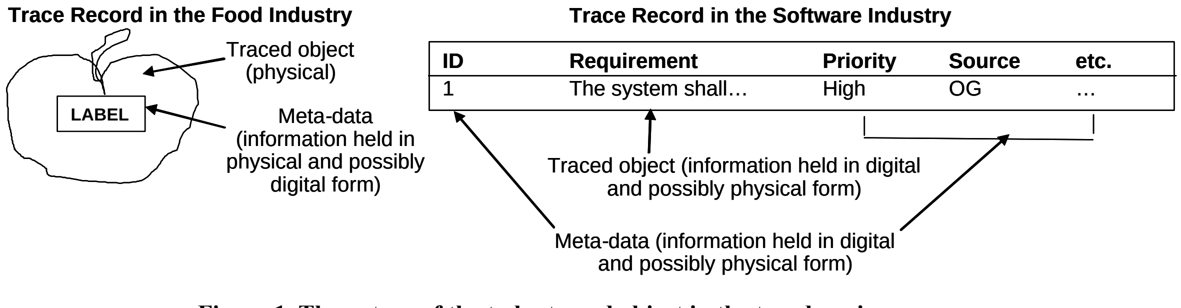

Trace Record in the Food Industry
Traced object (physical)
LABEL
Meta-data (information held in physical and possibly digital form)
Trace Record in the Software Industry
ID
Requirement
Priority
Source
etc.
1
The system shall…
High
OG
…
Traced object (information held in digital and possibly physical form)
Meta-data (information held in digital and possibly physical form)

Figure 1. The nature of the to-be-traced object in the two domains.

like explanation is, of course, clear when dealing with physical objects with known and controllable, or at least understood, possibilities for composition and decomposition.

In software development, all evidence post initial requirements elicitation may be indirect and so non-specific compared with some original documented requirement that the most appropriate definition of 'trace' becomes figurative:

d) *"A non-material indication or evidence of the presence or existence of something, or of a former event or condition."*

It is in this sense [(d)] that we talk about an idea 'disappearing without trace' or 'traces of ancestry'. In the absence of clear marks of its passing, any trace of a requirement can only be figurative. The key distinction is that no part of the antecedent exists in the consequent as there is no physical continuity of materials. This figurative use is common when 'track' is a verb:

e) *"To follow the course, development or history of."*

The definition [(d)] appears to apply to the to-be-traced object of software and to the meta-data about the objects under trace in both domains. Tracking and tracing become close to synonymous when using 'trace' in a literal sense, hence an alternative common definition of 'to trace':

f) *"To follow the footprints or traces of; especially to track by the footprints."*

Tracing ideas, concepts, even knowledge, is possible both literally and figuratively, although more likely the latter given the difficulty of demonstrating unequivocally that a mental mark has been made. A requirement 'trace', in its literal sense, demands not only a detailed explanation of the intermediate and final affects of the requirement but also a full record of all marks that it left at every transformation between documents and media. Any meaning for traceability in software development will remain essentially figurative until the potential transformations, and the associated relationships between to-be-traced objects, are understood and defined in a useable way [12].

7. TRACE RELATIONS

In the lifecycle of food, there are two basic types of trace relation, and these derive from the literal and figurative definitions of 'trace' that apply to it. The first type of relation is between the food product in some state X and the product in some state Y. This is necessarily a *temporal* linkage. Although temporal, it may be possible to reclaim the product in the original state X if it is preserved in its entirety in state Y (e.g. when an apple is merely packaged); the transformation is reversible if it is possible to recover the original apple by removing the packaging and then re-packaging it in a new way, though this is still effectively a forward step in time leading to a new state Z. Where the product in state X is materially altered in some way from state X to state Y (e.g. when an apple is used as an ingredient in an apple pie), then this transformation is non-reversible; it is not usually feasible to reconstitute the original apple and then reuse it in a different way. With this temporal relation there is always a detectable amount of the original food product across the two states. This provides for material continuity and a physical flow or path is apparent.

The second type of trace relation is concerned with the corresponding linkage that is formed between the two separate clusters of meta-data that are associated with the food product in its two states. This linkage is not subject to the same temporal constraints and is thus bi-directional; it is possible to read about the previous or subsequent contextual information at any point in time, and change it if desired, irrespective of whether or not it is possible to reclaim and make changes to the product item itself. This provides for an informational flow or conceptual path that is somewhat grounded in the physical one.

In the lifecycle of software, when a requirement description X is associated with another artifact Y in which the requirement may have been decomposed, refined, elaborated, evolved, explained, satisfied (or whatever link semantics are deemed relevant [19]), both X and Y still exist in their original states and have continued use or relevancy. The relation that is made is purely informational, as in the second type of trace relation above. It has nothing to do with material continuity across two states, but with people's perceptions of a connection between descriptions. It inter-relates two items that have unique identifiers in a way that seems to make sense to some person (or persons) at some moment in time and for some purpose. The 'trace' [(ii)] in the software world is formed by way of a plausible explanation that is made to bridge an inevitable gap between two different (or same) [(i)]'s. The implications are that there will not be a smooth transformation between traced objects as in the food industry because there is no physically continuous chain to hook into.

8. IF YOU JOIN THE DOTS…

Unlike the food lifecycle, the software lifecycle is characterized by a series of information points (e.g. requirements, design, code, etc.) that are first created and then jumped between, in such a way as to provide a reasonable explanation as to continuity. The expectation is that others can follow or reverse the path through the dots using the very same explanations and obtain an identical image. Traceability is a quality associated with movement and change, in time, space or form. In stasis it is merely inherent and becomes manifest only as a consequence of some action. This quality is acquired as a consequence of leaving some mark of passing presence, or because of the recognized potential so to do. Requirements are normally deemed to possess this quality only by virtue of this potential, always desired by their originators but never automatically realized without the intervention of others. Post elicitation much of requirements engineering concerns the nature of this intervention and its timing within the development process. The only alternative to this interventionalist, and essentially passive, approach is one based on ensuring that recognized marks, traces of presence, are made automatically and unavoidably, just as animals imprint snow or as they may be tagged in food industry processes.

9. TAKING A QUANTUM LEAP

"We tend to think of these spacetime histories as 'possible alternative classical trajectories' (in configuration space). The idea is that in the quantum world, instead of there being just one classical 'reality', represented by one such trajectory (one history), there is a great *complex superposition* of all these 'alternative realities' (superposed alternative histories)." [17]

In lay terms, ‘superposition’ means that an object possesses two or more values for some property at the same time (e.g. two positions in space or two points in time). In the food lifecycle, traced objects are only ever *perceived* to be in one state at any one time as the product moves forward linearly in normal time towards eventual consumption. It is not possible to go back to the past and recreate a new future state of an existing food product, except perhaps for the simplest of transformational steps, like packaging and un-packaging an apple pie. With the software lifecycle, the to-be-traced objects can be considered descriptions that together form an evolving model of the software to be built, at different levels of abstraction and from different viewpoints (following the position of [15]). It is therefore quite possible for any one to-be-traced object to exist in multiple states at any one time (i.e. at different levels of abstraction or within different models constructed from different viewpoints or embodying different components of the system). The necessity to be able to make a change to a requirement (obtained from the past) and propagate this forward to an existing design to change the present or future software product may mean that the traceability of software requirements is as straightforward as time travel!

The classic view of software development as producing a series of discrete information points representing an evolving model that we seek to establish a traceable path through may not only be limiting but may be flat-lining projects. In lay terms, the Copenhagen Interpretation states that a system stops being a superposition of states and becomes reduced to one or the other when an observation occurs [17]. In the software context, this would be each time a to-be-traced object is constructed and a trace relation is made between two such objects, both reliant on the knowledge of those performing the actions (see Figure 2). As with Newtonian physics and quantum physics, there are some things that cannot be explained or understood when restricted to the current way of thinking. Perhaps the analogy with quantum mechanics, well beyond the scope of this paper and utterly speculative, could yield some insight and trigger discussion into the complexity surrounding the tracing of abstract to-be-traced objects?

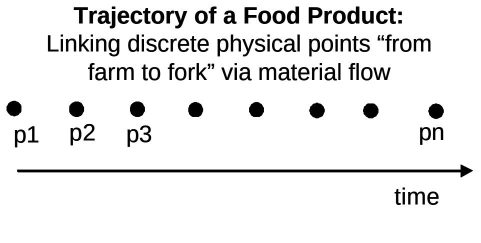
10. CONCLUSIONS

This paper suggests that there may be value in re-considering traceability at a more fundamental level, re-examining the nature of ‘traced’ or ‘to-be-traced’ objects, understanding what a ‘trace’ world of software. While perceived as costly and unattainable, there is really no clear explanation as to why this may be so. The paper proposes that there are lessons and insights that can be learned by looking at traceability in other domains. The food industry comparison makes for an appealing and tentative first contrast as it highlights why the traceability of requirements in software development may be harder than initially appears. It has highlighted an important distinction between 'literal trace' and 'figurative trace' that seems to have potential as a structure for thinking about what needs to be done. Are software to-be-traced objects actually traceable and so capable of being traced in a literal sense without physical counterpart? Can the trace relations ever be truly smooth, as in the food industry, or even bi-directional when people’s knowledge is what bridges the information gap?

Furthermore, traceability itself needs to be re-conceptualized as a measure of the *potential* for an object to be traced, making traceability specifically a quality or property of a to-be-traced object rather than an activity associated with the tracing or tracking process and stripped of any connotations thereof. Such objects may, by their nature, afford easy physical traces or may, by virtue of what affects them, leave little discernable impression on their environment. The activity of tracing is what requirements engineers or software engineers have to do, i.e. the physical tracking (creation of a ‘trace record’) of objects probably without intrinsic physical substance, only physical representation. Alternatively, or additionally, they have to render such objects more easily trackable (i.e. improve their traceability qualities either by some direct equivalent of physical tagging or better process understanding). Just as it is possible to measure software properties like reliability using a probability distribution and study its growth and decay over time, perhaps it makes sense to measure traceability growth and decay? This would permit the ‘goodness’ of the traceability to be measured with respect to some overarching traceability goal rather than leaving it as a poorly digestible and typically unknown mouthful.

11. ACKNOWLEDGEMENTS

We would like to thank the GCT’07 reviewers for their helpful pointers and constructive comments, including the reference to [8].


Trajectory of a Food Product:
Linking discrete physical points “from farm to fork” via material flow
p1
p2
p3
pn
time

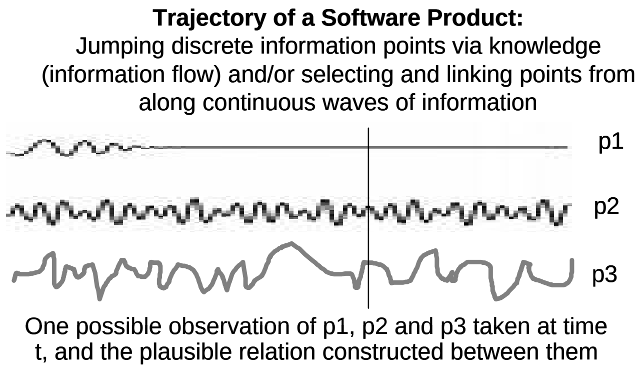

Trajectory of a Software Product:
Jumping discrete information points via knowledge (information flow) and/or selecting and linking points from along continuous waves of information
p1
p2
p3
One possible observation of p1, p2 and p3 taken at time t, and the plausible relation constructed between them

Figure 2. Are software to-be-traced objects continuous waveforms as well as discrete points? (Intentionally speculative!)

12. REFERENCES

[1] Arkley, P. and Riddle, S. Overcoming the Traceability Benefit Problem. *Proc. 13th IEEE International Requirements Engineering Conference*, IEEE Computer Society Press, Paris, France (August-September 2005), pp.385-389.

[2] Aubrey, D. Controlling the HMS Program through Managing Requirements. *Proc. 14th IEEE International Requirements Engineering Conference*, IEEE Computer Society Press, Minneapolis/St. Paul, MN (September 2006), pp.222-227.

[3] Byrne, D. The Regulation of Food Safety and the Use of Traceability/Tracing in the EU and USA: Convergence or Divergence? *Food Safety Conference*, Washington, DC, 19th March, 2004. (Available from 2004 Speeches and Press Conferences, European Union: Delegation of the European Commission to the USA.)

[4] Cleland Huang, J., Dekhtyar, A. and Huffman Hayes, J. (Eds.) *Grand Challenges in Traceability*. Center of Excellence for Traceability Technical Report COET-GCT-06-01, University of Kentucky, September 2006.

[5] The European Parliament and the Council of the European Union. *Regulation (EC) No 178/2002 of the European Parliament and of the Council of 28 January 2002*. Section 15, 28, 29 and Article 18, Official Journal of the European Communities.

[6] European Commission. *From farm to fork: Safe food for Europe's consumers*. Europe on the Move Series, Luxembourg: Office for Official Publications of the European Communities, 2004 (ISBN 92-894-7772-5).

[7] Food and Drug Administration, U.S. Department of Agriculture Centers for Disease Control and Prevention. *Guide to Minimize Microbial Food Safety Hazards for Fresh Fruits and Vegetables*. IX: 'Traceback', October 26th, 1998.

[8] Food and Drug Administration, U.S. Department Of Health and Human Services, Center for Devices and Radiological Health, Center for Biologics Evaluation and Research. *General Principles of Software Validation; Final Guidance for Industry and FDA Staff*. January 11th, 2002.

[9] Golan, E., Krissoff, B. and Kuchler, F. Food Traceability: One Ingredient in a Safe and Efficient Food Supply. *Amber Waves: The Economics of Food, Farming, Natural Resources, and Rural America*, U.S. Department of Agriculture, Economic Research Service, April 2004.

[10] Golan, E., Krissoff, B., Kuchler, F., Calvin, L., Nelson, K. and Price, G. *Traceability in the U.S. Food Supply: Economic Theory and Industry Studies*. AER-830, U.S. Department of Agriculture/ Economic Research Service, March 2004.

[11] Gotel, O.C.Z. and Finkelstein, A.C.W. An Analysis of the Requirements Traceability Problem. *Proc. 1st IEEE International Conference on Requirements Engineering*, IEEE Computer Society Press, Colorado Springs, CO (April 1994), pp.94-101.

[12] Gotel, O.C.Z. and Morris, S.J. Crafting the Requirements Record with the Informed Use of Media. *Proc. 1st International Workshop on Multimedia Requirements Engineering* (with 14th IEEE International Requirements Engineering Conference), Minneapolis/St. Paul, MN, September 12th, 2006.

[13] Huffman Hayes, J., Dekhtyar, A. and Osborne, J. Improving Requirements Tracing via Information Retrieval. *Proc. 11th IEEE International Requirements Engineering Conference*, IEEE Computer Society Press, Monterey, CA (September 2003), pp.138-147.

[14] International Organization for Standardization. *International Vocabulary of Basic and General Terms in Metrology (VIM)*. ISO VIM (DGUIDE 99999), Definition 2.24 (6.10) 'metrological traceability', revision of 1993 edition, 2004.

[15] Jackson, M.A. The General and the Particular. *Challenges and Strategies for Research in Systems Development: Papers from a Conference held at Georgia State University,* November 1988; Cotterman, W. W. and Senn, J. A. (Eds.), John Wiley & Sons, 1992, pp.33-40.

[16] *The Oxford English Dictionary*. Online Version, Oxford University Press, http://www.oed.com (accessed January 2007).

[17] Penrose, R. *The Road to Reality*. London: Cape, 2004, p.667 and p.783.

[18] Pohl, K. *Process-Centered Requirements Engineering*. RSP, Somerset, U.K. (John Wiley & Sons, Ltd., U.K.), 1996.

[19] Ramesh, B. and Jarke, M. Toward Reference Models for Requirements Traceability. *IEEE Transactions on Software Engineering, 27, 1* (January 2001), pp.58-93.

[20] TEFSE 2002. *Proceedings of 1st International Workshop on Traceability in Emerging Forms of Software Engineering* (with 17th IEEE International Conference on Automated Software Engineering), Edinburgh, U.K., September 28th, 2002.

[21] TEFSE 2003. *Proc. 2nd International Workshop on Traceability in Emerging Forms of Software Engineering* (with 18th IEEE International Conference on Automated Software Engineering), Montreal, Canada, October 7th, 2003.

[22] TEFSE 2005. *Proc. 3rd International Workshop on Traceability in Emerging Forms of Software Engineering* (with 20th IEEE International Conference on Automated Software Engineering), Long Beach, CA, November 8th, 2005.