SOEN 6431
SOFTWARE
MAINTENANCE
AND
PROGRAM
COMPREHENSI
ON

Week 2 - Software Traceability

UNIVERSITÉ Concordia UNIVERSITY

GINA CODY
SCHOOL OF ENGINEERING
AND COMPUTER SCIENCE

Juergen Rilling, PhD
Professor
Department of Computer Science
and Software Engineering

MAILING ADDRESS
1455 DE MAISONNEUVE BLVD. W., EV-3.211
MONTREAL, QUEBEC, CANADA H3G 1M8
CONCORDIA.CA

514-848-2424 ext. 3016
juergen.rilling@concordia.ca
concordia.ca/ginacody
STREET ADDRESS
1515 Ste. Catherine St. W., EV-3.211

Traceability …

already a major factor in our daily lives

Recalls and safety alerts

Canada

White-Rodgers Gen II Intellivent natural gas and propane control valves installed on certain models of Giant and Gemco natural gas and propane water heaters recalled due to potential carbon monoxide leak

Report a Concern

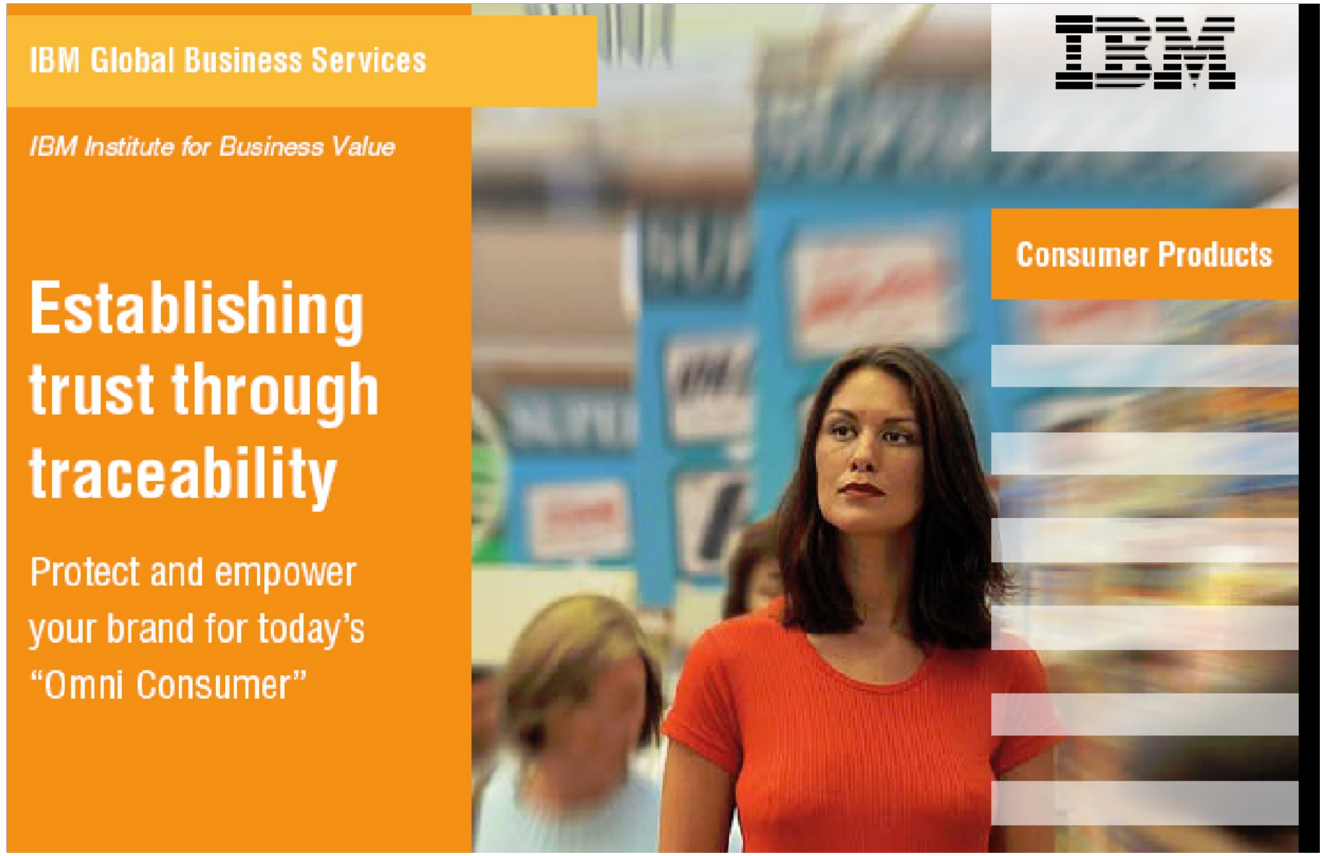
Summary

Affected products

Affected products

Video Food Traceability

Traceability … already a major factor in other domains



To ensure safety of the food supply and other consumer products, as well a enhance credibility with consumers, consumer products companies can turn to full value traceability to track product ownership throughout the supply chain.

Centers for Disease Control and Prevention

Search COVID-19

COVID-19

Your Health | Vaccines | Cases & Data | Work & School | Healthcare Workers | Health Depts | Scien

Your Health

About COVID-19

Variants of the Virus

Symptoms

Testing

Testing for COVID-19

Test for Current Infection

Test for Past Infection

Self-Testing

Contact Tracing

How to Talk to Your Close Contacts

Contact Tracing Steps

Prevent Getting Sick

To maximize protection from the Delta variant and prevent possibly spreading it to others, get vaccinat
you can and wear a mask indoors in public if you are in an area of substantial or high transmission.

Contact Tracing

Contact tracing is key to slowing the spread of COVID-19 and helps protect you, your family
community.

Updated Feb. 25, 2021 | Languages | Print

Contact tracing slows the spread of COVID-19

Contact tracing helps protect you, your family, and your community by:

- Letting people know they may have been exposed to COVID-19 and should monitor their health for signs and symptoms of COVID-19.
- Helping people who may have been exposed to COVID-19 get tested.
- Asking people to self-isolate if they have COVID-19 or self-quarantine if they are a close contact.

On This Page

What you can expe
you're:

Around someone di
COVID-19

Waiting for a COVID-

Diagnosed with COV

ot ask you for:

Canada

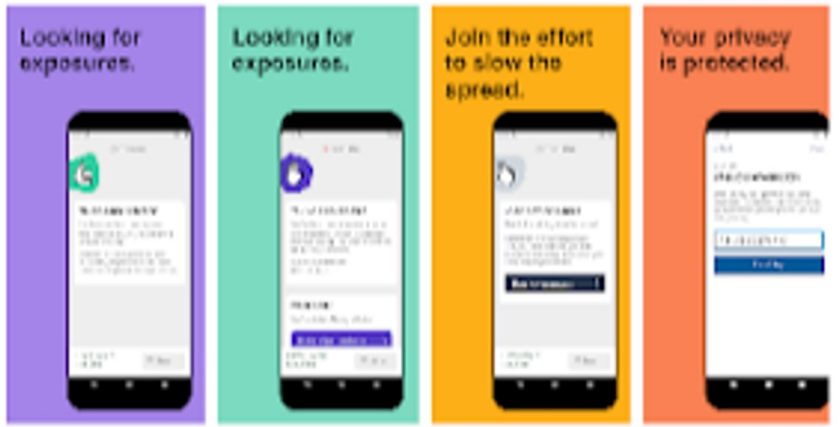
COVID Alert - Let's protect each other

Health Canada | Santé Canada Health & Fitness ★★★★☆ 3,022

Everyone

This app is available for some of your devices

Installed



Together, let's slow the spread of COVID-19. Canada's COVID Alert app notifies you if someone you were near in the last 14 days tells the app they tested positive.

COVID Alert uses Bluetooth to exchange random codes with nearby phones. It does not use or access any location data. COVID Alert works by determining how far away other phones are by the strength of their Bluetooth signal.

Several times a day, COVID Alert checks a list of codes from people who tell the app they tested positive. You'll get a notification if it finds you received one of the positive codes.

And more recently….

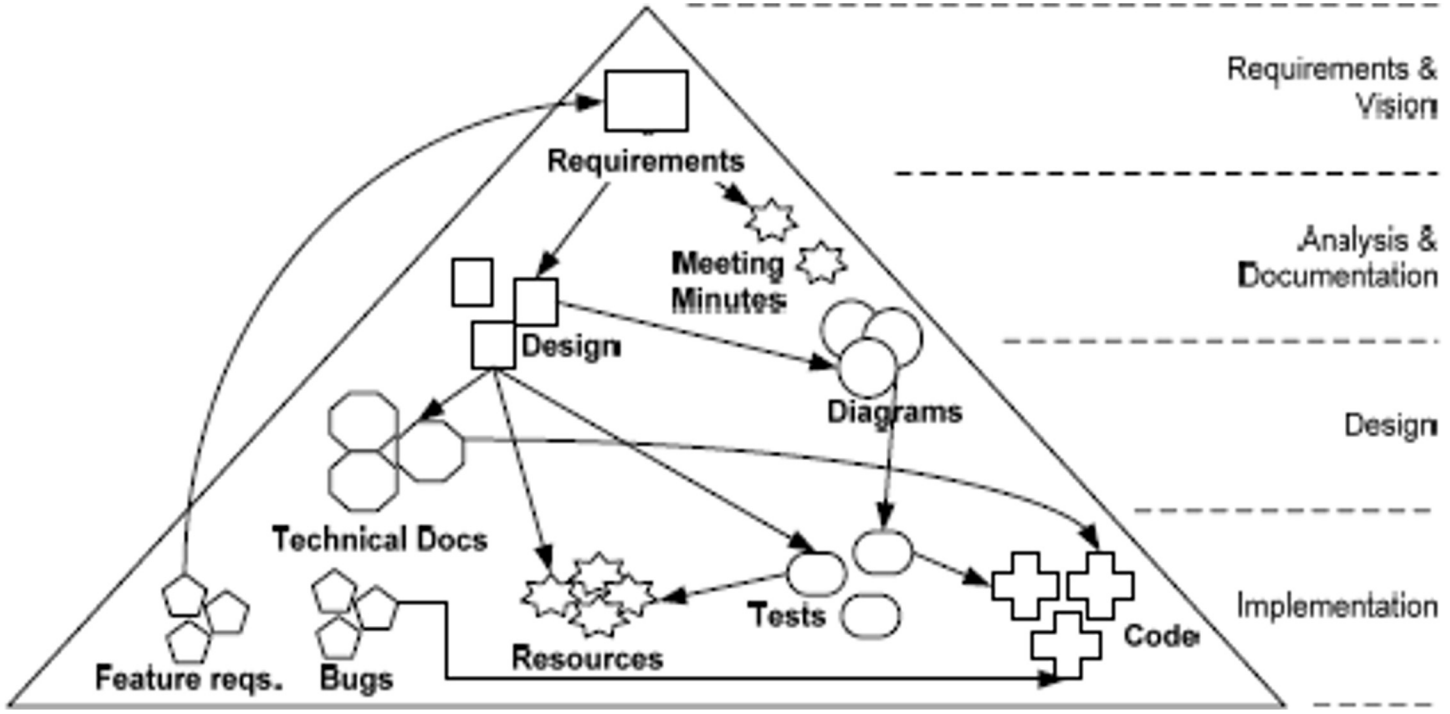
You can't manage what you can't trace.

[Watkins and Neal, 1994]

Robert Watkins, Mark Neal: Why and How of Requirements Tracing. IEEE Software 11(4): 104-106 (1994)

Definitions: Software Traceability

- **"Traceability […] a document in question is in agreement with a predecessor document to which it has a hierarchical relationship. [DoD 1988]**
- **A Software Requirements Specification (SRS) is traceable if the origin of each of its requirements is clear and if it facilitates the referencing of each requirement in future development or enhancement documentation." [IEEE 1998].**



[IEEE 1998] IEEE, Std. 830: Guide to Software Requirements Specification," 1998.
[DoD 1988] DoD, Std-2167A: U.S. Department of Defense Military Standard: Defense System Software Development, Washington, D.C. 1988.

Traceability gives essential assistance in understanding the relationships that exist within and across software requirements, design, and implementation. ***[Palmer, 2000]***

J. D. Palmer. Traceability. In R. H. Thayer and M. Dorfman, editors, Software Requirements Engineering, Second Edition, pages 412-422. IEEE Computer Society Press, 2000.

Traceability Quotes (1)

Requirements traceability refers to the ability to describe and follow the life of a requirement, in both forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of ongoing refinement and iteration in any of these phases)".[1]

A software requirements specification is traceable if the origin of each of its requirements is clear and if it facilitates the referencing of each requirement in future development or enhancement documentation.[2]

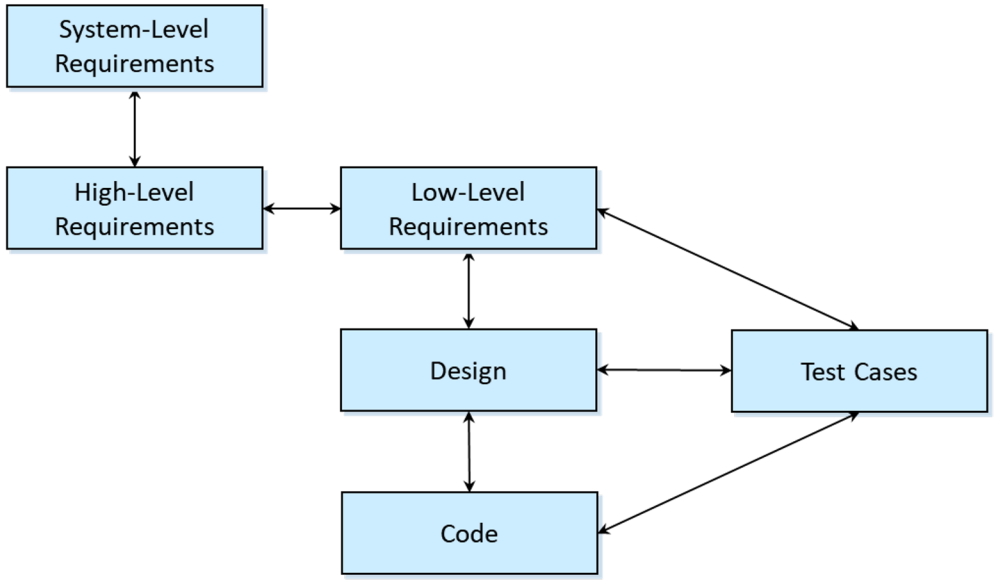
Traceability gives essential assistance in understanding the relationships that exist within and across software requirements, design, and implementation.[3]

A link or relationship defined between entities.[4]

[1] Gotel & Finkelstein, 1994; [2] IEEE Standard 830-1998; [3] Palmer, 2000; [4] Watkins and Neal, 1994

DOD-178B

At all stages of the process, traceability is required.



Traceability Quotes (2)

- Traceability is often mandated by contracts and standards.[1]
  - E.g., military and aerospace
- One cannot manage what cannot be traced.[2]
- Traceability information helps assess the impact of changes to requirements, connecting these requirements as well as requirements for other representations of the system.[3]
- Traceability is a property of a system description technique that allows changes in one of the three system descriptions – requirements, specifications, implementation – to be traced to the corresponding portions of the other descriptions. The correspondence should be maintained through the lifetime of the system.[4]

[1-2] Watkins and Neal, 1994; [3] Kotonya and Sommerville, 1998; [4] Greenspan, McGowan, 1978

Importance of Traceability (1)

- Requirements cannot be managed effectively without requirements traceability

A requirement is traceable if you can discover:

- who suggested the requirement, why the requirement exists,
- what requirements are related to it and
- how that requirement relates to other information such as systems designs, implementations and user documentation

Importance of Traceability (2)

Benefits of traceability

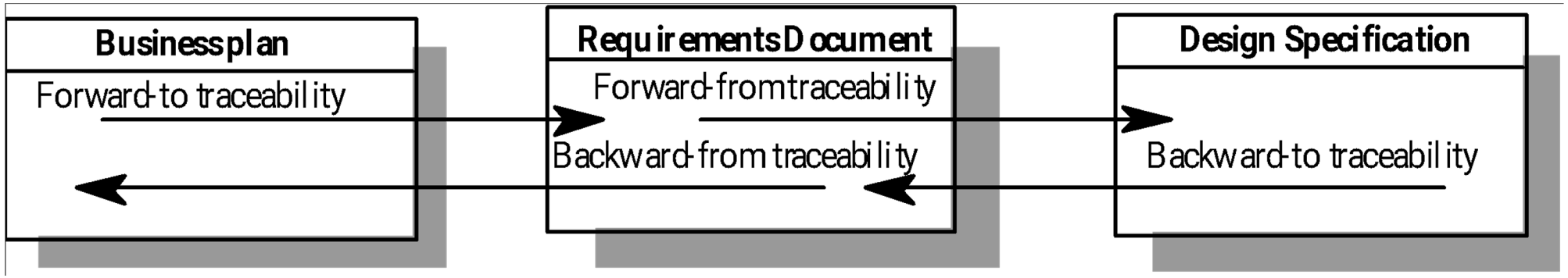
- Prevents losing knowledge
- Supports the verification process (certification, localization of defects)
- Impact analysis
- Change control
- Process monitoring (e.g., missing links indicate completion level)
- Improved software quality (make changes correctly and completely)
- Reengineering (define traceability links is a way to record reverse engineering knowledge)
- Reuse (by identifying what goes with a requirement: design, code…)
- Risk reduction (e.g., if a team member with key knowledge leaves)

Traceability Difficulties

- Various stakeholders require different information
- Huge amount of requirements traceability information must be tracked and maintained
- Manual creation of links is ***very*** demanding
  - Likely the most annoying problem
- Specialized tools must be used
- Integrating heterogeneous models/information from/to different sources (requirements, design, tests, code, documentation, rationales…) is not trivial
- Requires organizational commitment (with an understanding of the potential benefits)

Backward and Forward Traceability (1)

- **Backward traceability**
  - To previous stages of development
  - Depends upon each element explicitly referencing its source in earlier documents
- **Forward traceability**
  - To all documents spawned by a document
  - Depends upon each element in the document having a unique name or reference number



Source of figure: Kotonya and Sommerville

Backward and Forward Traceability (2)

Top to bottom from requirements' point of view

- **Forward-to traceability**
  - Links other documents (which may have preceded the requirements document) to relevant requirements
  - Help validation
  - Help evaluate which requirements are affected by changes to users' needs
- **Forward-from traceability**
  - Links requirements to the design and implementation components
  - Help assure that all requirements have been satisfied

Backward and Forward Traceability (3)

Bottom to top from requirements' point of view

- **Backward-to traceability**
  - Links design and implementation components back to requirements
  - Help determine why each item is designed/implemented
- **Backward-from traceability**
  - Links requirements to their sources in other documents or people
  - Help validation
  - Help evaluate how changes to requirements impact stakeholders needs

Representation – Traceability Table

- Show the relationships between requirements or between requirements and other artifacts
- Table can be set up to show links between several different elements
- Backward and forward traceability
- Difficult to capture different types of links

User Requirement	Functional Requirement	Design Element	Code Module	Test Case
UC-28	catalog.query.sort	Class Catalog	catalog.sort()	search.7 search.8
UC-29	catalog.query.import	Class Catalog	catalog.import(), cat catalog.validate()	search.12 search.13 search.14

Representation – Traceability Matrix

Depends-on

	R1	R2	R3	R4	R5	R6
R1			*	*		
R2					*	*
R3				*	*	
R4		*				
R5						*
R6						

Define links between pairs of elements

E.g., requirements to requirement, use case to requirement, requirement to test case…

Can be used to defined relationships between pairs

E.g., specifies/is specified by, depends on, is parent of, constrains…

More amenable to automation than traceability table

Representation – Traceability List

Requirement	Depends-on
R1	R3, R4
R2	R5, R6
R3	R4, R5
R4	R2
R5	R6

Traceability matrices become more of a problem when there are hundreds or thousands of requirements as the matrices become large and are sparsely populated

A simplified form of a traceability matrix may be used where, along with each requirement description, one or more lists of the identifiers of related requirements are maintained

Traceability Planning

- Planning traceability?
  - Yes, just like any other project!
- Who are the stakeholders?
- What are the needs (analysis, reports…)?
  - Useful, measurable, feasible objectives
- Definition of links and attributes
  - Can some be inferred automatically?
- Process (who collects and when to collect traceability information)
  - Roles, access
  - Data/link input and updates
  - Exceptional situations (e.g., lack of time)
- Representations, queries, tools
- Traceability policies define what and how traceability information should be maintained

Factors to Consider during Planning (1)

Number of requirements

The greater the number of requirements, the more the need for formal traceability policies

Expected system lifetime

More comprehensive traceability policies should be defined for systems which have a long lifetime

Maturity level of organization

Detailed traceability policies are more likely to be implemented and used properly in a cost-effective way in organizations which have a higher level of process maturity

Size of project and team

The larger the project or team, the greater the need for formal traceability policies

Factors to Consider during Planning (2)

Link Source Object Type	Link Target Object Type	Information Source
System requirement	Software requirement	System engineer
Use case	Functional requirement	Requirements analyst
Functional requirement	Functional requirement	Requirements analyst
Functional requirement	Test case	Test engineer
Functional requirement	Software architecture element	Software architect
Functional requirement	Other design elements	Designer or Developer
Business rule	Functional requirement	Requirements analyst

Type of system

Critical systems such as hard real-time control systems or safety-critical systems need more comprehensive traceability policies than non-critical systems

Additional constraints from customer

E.g., compliance to military standard

Traceability links should be defined by whoever has the appropriate information available

Modeling Traceability

The types of links to use (and their attributes) must be defined for different types of requirements

It is a design problem!

- ***May be modeled with a UML class diagram (metamodel)***

Object types (classes)

Object attributes (attributes)

Structure of folders, modules, objects

- Stereotypes, composition…

Link types (associations)

- Satisfies, tests, refines, contains, contributes to, threatens, justifies…

Link attributes (association classes)

…

Types of Traceability Links

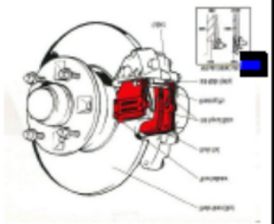
Note the types of links in the previous examples, as well as the types of objects they relate

Satisfies, Tests
Refines,
References,
Contains...

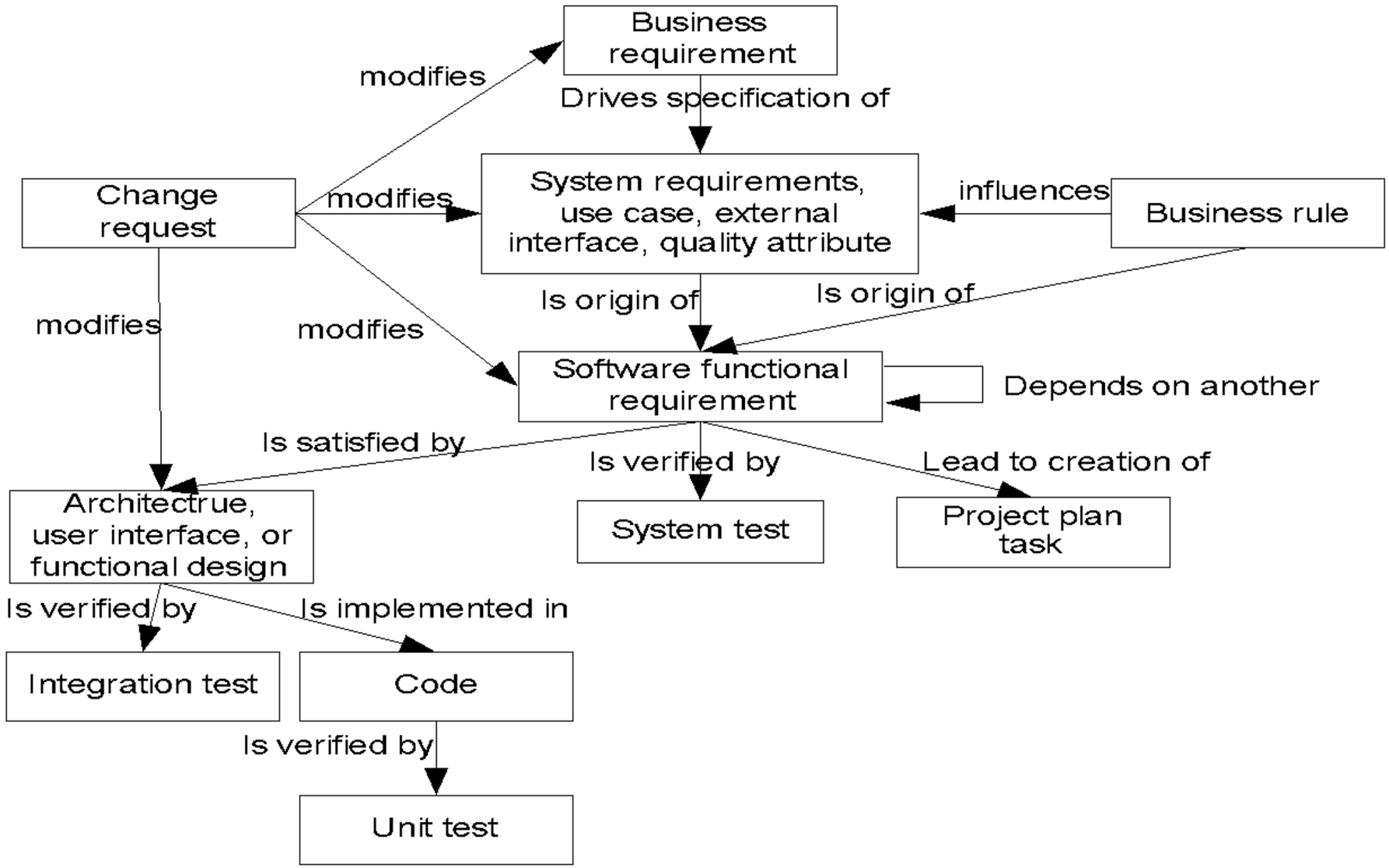
Others could be created

Contributes,
Contradicts,
Justifies, Depends on...

Requirements	Design	Code (software)	Documentation	Test cases

....
5.1. Braking distance <50 m when speed ~90 km/h
5.2. Absorbers should be electronically controlled.
		public ABS control(String args[])
		throws Exception
		{
		Class c = null;
		if (args.length == 1)
		{
			...
			Braking: The driver should push brakes sharply to the utmost.
			...
				Braking test:
				- on dry asphalt;
				- on slippery roads
				- on bumpy roads

Other Example of Traceability Links



Cardinality of Traceability Links

Depending on the traceability information, the cardinality of traceability links may be

- One-to-one
  - e.g., one design element to one code module
- One-to-many
  - e.g., one functional requirement verified by multiple test cases
- Many-to-many
  - e.g., a use case may lead to multiple functional requirement, and a functional requirement may be common to several use cases

The early days of traceability (1970s)

- Dating back to the 70's
  - Pierce introduced a requirements tracing tool as a way to build and maintain a requirements database and facilitate. requirements analysis and system verification and validation for a large Navy undersea acoustic sensor system. [Pierce 1978]
  - Links are manually maintained.
  - Underlying basic concept was some form of requirement traceability (matrices).

Unique Requirement ID	Requirement Description	Design Reference	Module / Configured Item Reference	Release Reference	Test Script Name/Step Number Reference

General requirements traceability view

SRS Requirement	Design Component(s) (CSC's)
3.2.1.1	1.0, 1.1
3.2.1.2	1.0, 1.1
3.2.1.3	1.0, 1.1
3.2.1.4	1.0, 1.1

Requirements -> Design

Design Component(s) (CSC's)	SRS Requirement
1.0	3.2.1.1, 3.2.1.2, 3.2.1.3, 3.2.1.4, 3.2.1.5, 3.2.1.6, 3.2.1.7, 3.2.1.8, 3.2.1.9, 3.2.1.10, 3.2.1.11, 3.2.1.12, 3.2.1.13, 3.2.1.15, 3.2.1.16, 3.2.1.17, 3.2.2.1, 3.2.2.2, 3.2.2.3, 3.2.2.4, 3.2.2.5, 3.2.2.6, 3.2.2.7, 3.2.2.8, 3.2.3.1, 3.2.3.2, 3.2.4.1, 3.2.4.2, 3.2.4.4, 3.2.4.5, 3.2.5.2, 3.2.5.3, 3.2.5.4, 3.2.5.5, 3.2.5.6, 3.2.5.7, 3.2.5.8, 3.2.5.9, 3.2.5.10, 3.2.5.11, 3.2.5.12, 3.2.5.13, 3.2.5.14, 3.2.5.15, 3.2.5.16, 3.2.5.17, 3.2.5.18, 3.2.6.1, 3.2.7.1, 3.2.7.2, 3.2.7.3, 3.2.7.4, 3.2.8.1, 3.2.8.3, 3.3.1

Design -> Requirements

NASA - ISD Requirements Traceability Matrix Guidelines (effective date June 2005 – June 2010)
(http://software.gsfc.nasa.gov/AssetsApproved/PA2.2.1.3.doc

[Pierce 1978] R. Pierce, "A Requirements Tracing Tool," Proc. Software Quality Assurance Workshop Functional and Performance Issues, 1978.

Early days of traceability (late 1980s – early 90s)

- Attempts were made to model and standardize traceability.
  - Model/Schema driven approaches to automatically generate and maintain traceability links.
  - Mainly driven by CASE tools that provided specialized solutions
    - Focus initially on requirements traceability matrices.
    - Attempts were made to facility the traceability among other software lifecycle products, including knowledge. [Ramesh 1992]
    - Support for other traceability applications like change impact analysis, backward tracing, verification and validation

Early days of traceability (late 1980s – early 90s)

- "Future systems may incorporate these….specialized tools … into the CASE environment to take advantage of access to the data dictionary for traceability, configuration control, and integrated revision cycles". [Tamanaha 1989]

- IBM/AD Cycle – Repository manager [Mercurio 1990]

Cross life-cycle tools. Whereas some application development tools provide functional support for a single phase of the application development life cycle, management of projects, the process itself, documentation, and changes (impact analysis) extend across all phases.

Having consistency and integration at the end-user interface is a major factor in the structure of the AD/Cycle architecture.

[Tamanaha 1989] Tamanaha, D.Y.; Wenjen, W.C.; Patel, B.K., "The application of CASE in large aerospace projects", Aerospace Applications Conference, 1989. Digest., 1989 IEEE 12-17 Feb. 1989 Page(s):18 pp.

AD/Cycle strategy and architecture by V. J. Mercurio, B. F. Meyers, A. M. Nisbet, G. Radin, IBM Journal of Research and Development Volume 29, Number 2, Page 170 (1990)

The early days of traceability (Mid 1990s)

Existing traceability approaches are no longer applicable !

- Shift in programming (Functional vs. OO)
- Shift in processes and models to be applied (Waterfall vs. RUP)
- Shift in the hardware paradigms (Mainframe vs. PCs)
- Existing tools are no longer a valid option.

New traceability approaches are needed !

- Due to the ever changing paradigms, there is even more of a need to create and re-create traceability. In particular the use of iterative and agile development processes.
- New techniques, tools and approaches are needed to support these new paradigm.

Key Developments From the Late 90s - Present

- COTS, outsourcing, among others are causing a change of system ‘ownership’
  - Limited expertise with system.
  - No or only partial access to relevant artifacts.
- Requirements/design/code traceability is difficult enough in conventional systems but much harder in
  - Heterogeneous systems.
  - Requiring tracing interactions, requirements, source code and artifacts among subsystems and locations.

Key Developments From the Late 90s - Present

Widespread use of Agile and Iterative development processes.

- “The lack of models. E.g. in XP only 10-20 minutes of fast sketching on a white board is common. This allows only for a rough outline and may not reflect the real implementation.
- A lot of knowledge at this stage is only kept as group knowledge among developers. XP promotes minimal documentation and compensates this with an increased level of communication (team and customer).
- But this information is lost after the project ends”. [Larman 2003]

Key Developments From the Late 90s - Present

Open source community makes new, large information sets are available to the research community

- CVS with build and change histories
- Bug report databases (e.g. Bugzilla)
- Electronic documents (e.g. e-mails, discussion, support groups, etc.)

Focus on mining these large data set to re-establish links for different maintenance aspects, including

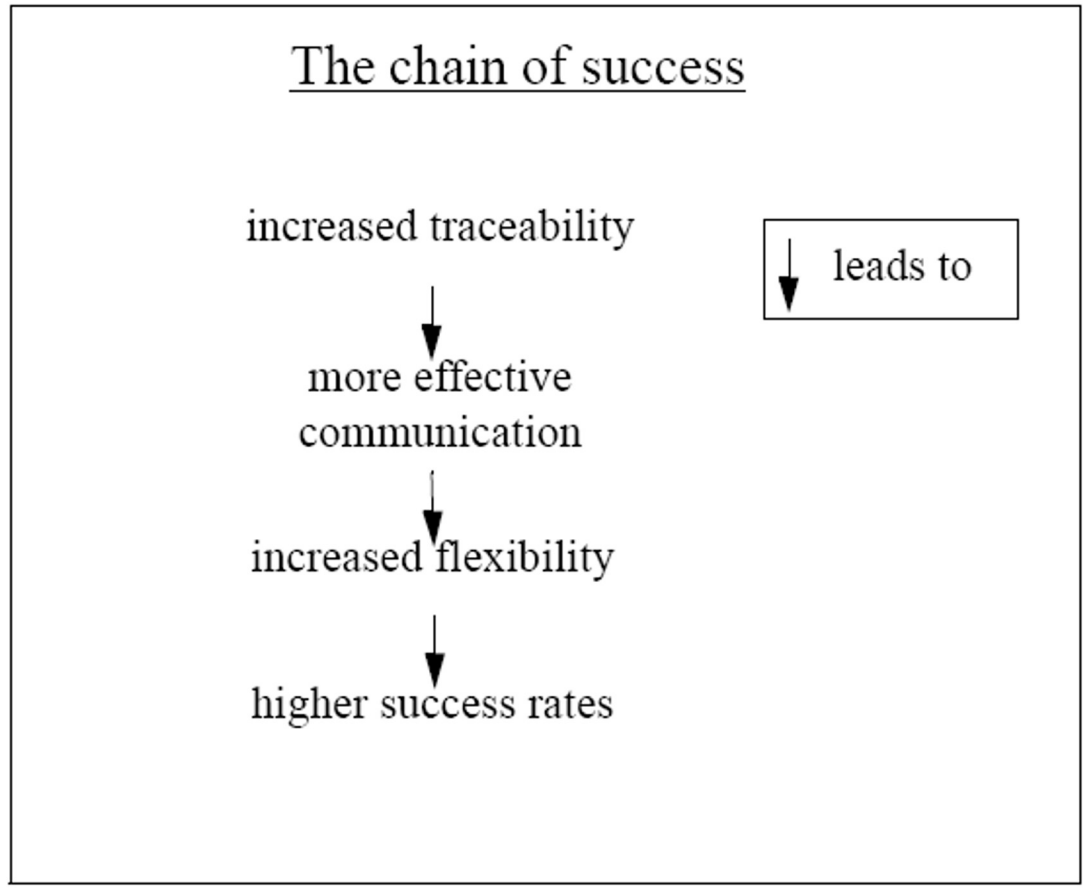
- Fault prediction
- Impact analysis
- Quality evaluation
- Restructuring

Key Developments From the Late 90s - Present

- The Internet as medium for sharing information and collaboration
  - Resulting in a cultural change – Sharing information and ideas becomes part of the daily lives (blogging, online communities, Wikis, etc.) and may need to be traced back to other software artifacts
- A community based approach to tool development, support different aspects of traceability and related analysis techniques
  - Eclipse
  - Allows specialized plug-in development to support traceability and their integration through a common framework.

Key Developments From the Late 90s - Present

- Traceability and process improvement.
  - Traceability becomes a important quality factor.
  - Integrates traceability within traditionally separate organizational functions set process improvement goals and priorities, provide guidance for quality processes
- Information retrieval becomes widely accepted as a valid approach to recover traceability links from different artifacts, including bug-reports, etc.
  - Probabilistic Information Retrieval (IR) and vector retrieval
  - Latent Semantic Indexing (LSI) or also known as LSA (Latent Semantic Analysis)
- Quality measurements are introduced, namely recall and precision are introduced to evaluate the quality of the links.


The chain of success
increased traceability
more effective communication
increased flexibility
higher success rates
leads to

Software Traceability

To support:

- (Co-)evolution
  - Change impact analysis
  - Maintenance
  - Reengineering
- Certification
- Project tracking
- Reuse
- Risk reduction
- Testing

Conclusions:

- Requirements often evolve due to environmental changes
- Suitability for evolution: quality, volatility, dependencies, inconsistency
- Evolution:
  - Continuous change and growth
  - System architecture is “almost” stable
- Co-evolution
  - Need for backward and forward traceability

Group Discussion items

- Agile development versus Traceability?
  - Back to the Waterfall model !!!??
- Should we mandate/legislate traceability?
- Traceability vs. privacy and security?
- Open source and traceability?
- Incomplete documents/artifacts
  - No complete requirements
  - No source code (COTS)

Project:

Project:

WHO IS FAMILIAR WITH TRACKING, HEALTH APPLICATIONS SUCH AS:

STRAVA, GOOGLE FIT, SPORTSTRACKER, OPENTRACKS…

Project:

WHO IS FAMILIAR WITH GITHUB?

WHO IS FAMILIAR WITH JAVA AND ANDROID?

Project:

GOOD NEWS – YOU WILL HAVE AN OPPORTUNITY TO LEARN ALL OF THEM !!