Requirements Testing: Turning Compliance into Commercial Advantage

Mike Bartley,

Test and Verification Solutions

Agenda

Some theory

- Requirements management
- Mapping requirements to tests

Some practice

Some reflection

- Using SQL
- Recording test results

- Business advantages

So what about software requirements?

- **Software requirements are much more complex**

  - **missed deadlines**
  - **exceeded budget**
  - **inability to meet project reqs**

- **Poor and Changing Requirements have been the main cause of project failures for years**
  - Bull 1998 (203 interviews) major causes of failure:
    - Highest = breakdown in communications (57%)
  - Chaos Report 1995 (308 managers):
    - Highest = incomplete requirements (13.1%)

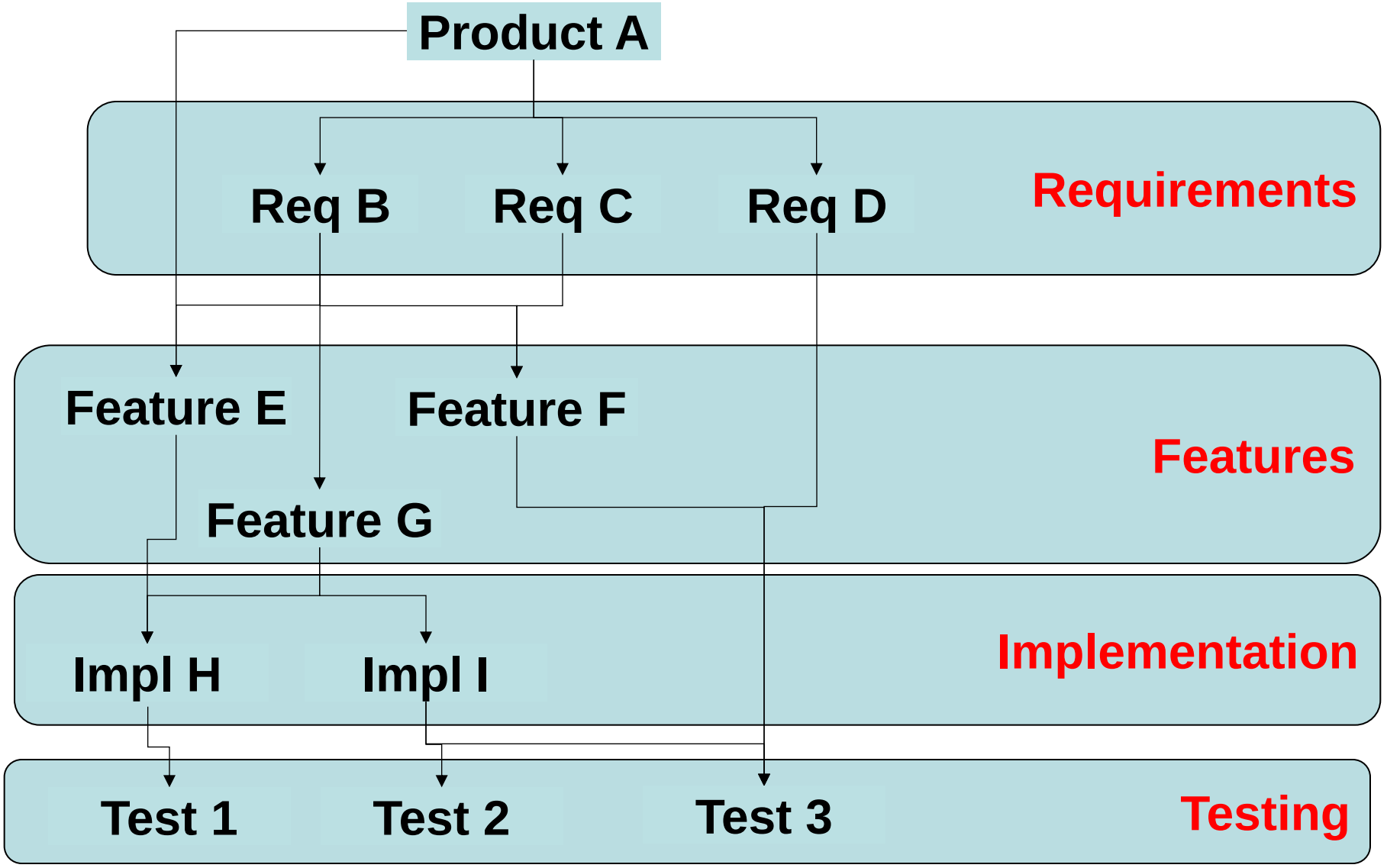
Agile attempts to solve this issue with requirements

Capturing software requirements

- **Over the years we have learned many ways to capture requirements:**
  - Documents (User / Marketing Requirements Documents)
  - Use Cases or Stories
  - Specification by Examples
  - Tests as Specifications
  - Formal specifications
- **But how do we make sure**
  - Requirements are implemented
  - And tested

Some industries Mandate this

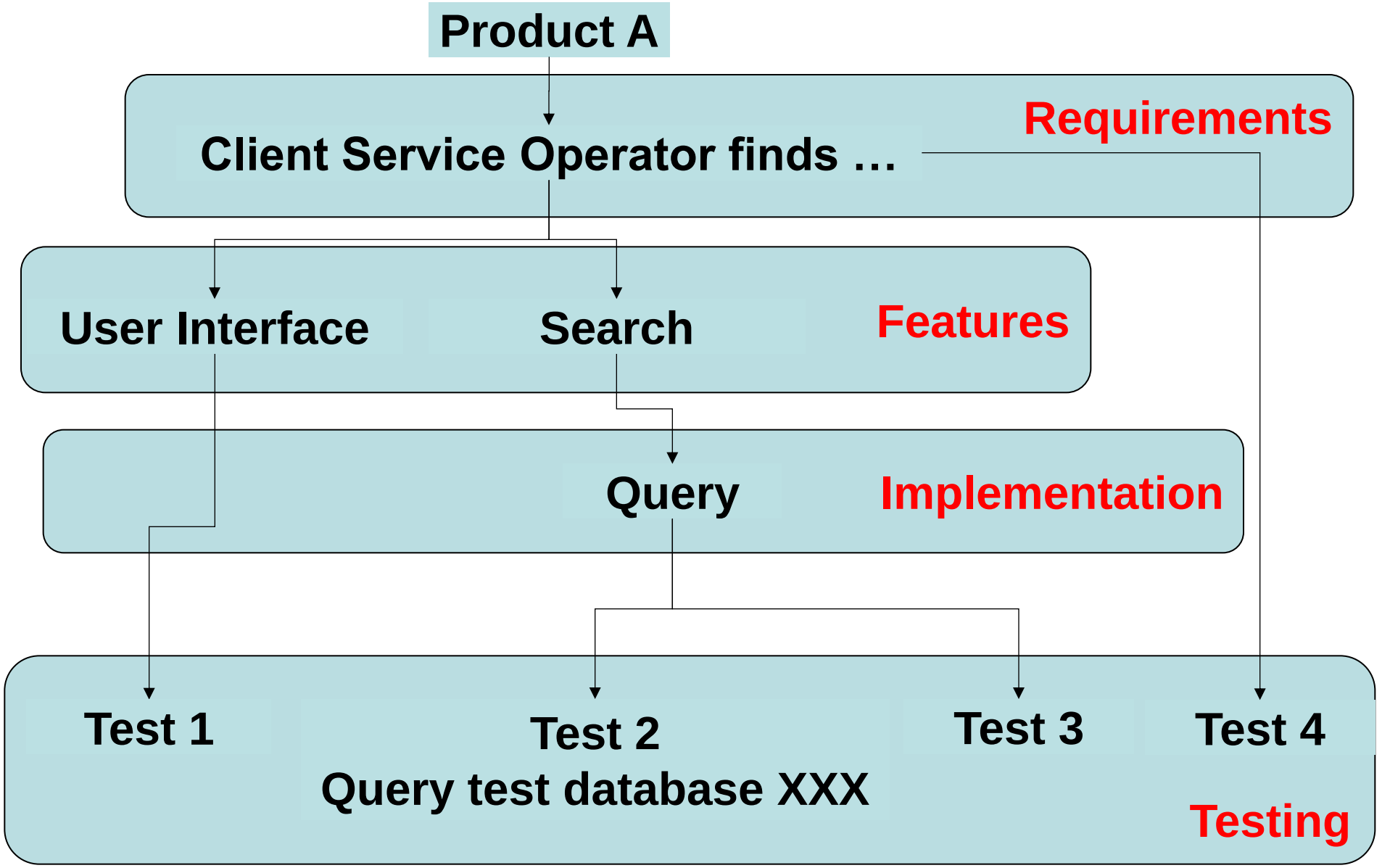
Ensuring Requirements are Implemented and Tested



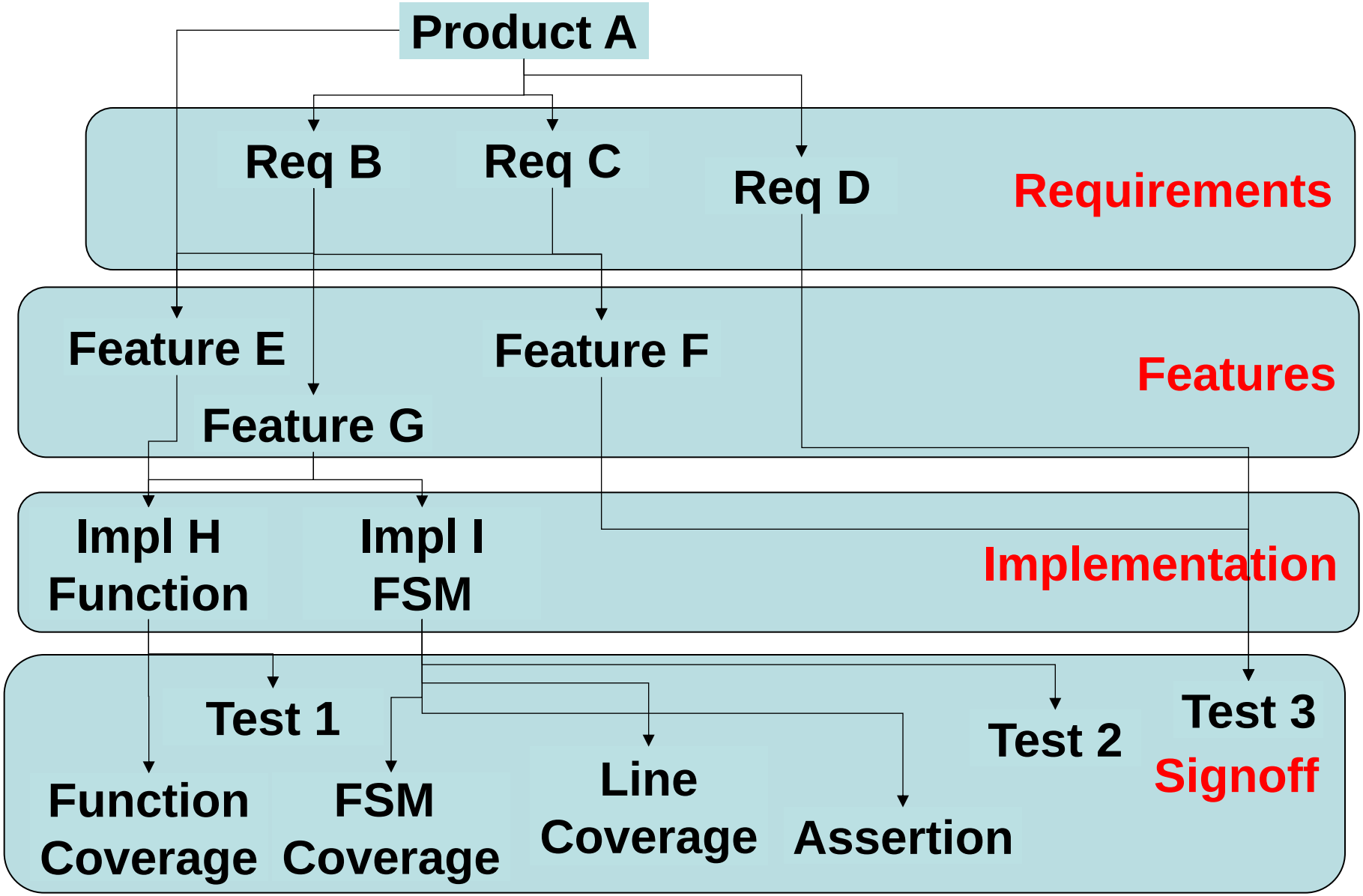
Example

- **Requirement**
  - Client Service Operator finds all current clients in the system where their total value of sales between 2 specified dates is above a specified value
- **Features**
  - User (with suitable privilege) can search system for current clients where total value of sales between 2 specified dates is above a specified value
  - User Interface
    - allows a user to select the report, select date range etc
- **Implementation Aspects**
  - For every transaction in the system the following data must be stored
    - Revenue value, Date and Client reference
  - Need a query to find clients where total transaction value between 2 dates is above specified value
  - Authorisation, Exceptions, Etc.
- **Tests**
  - Query test database XXX with dates “d1/m1/y1” and “d2/m2/y2”. Expected result = “client1”,...

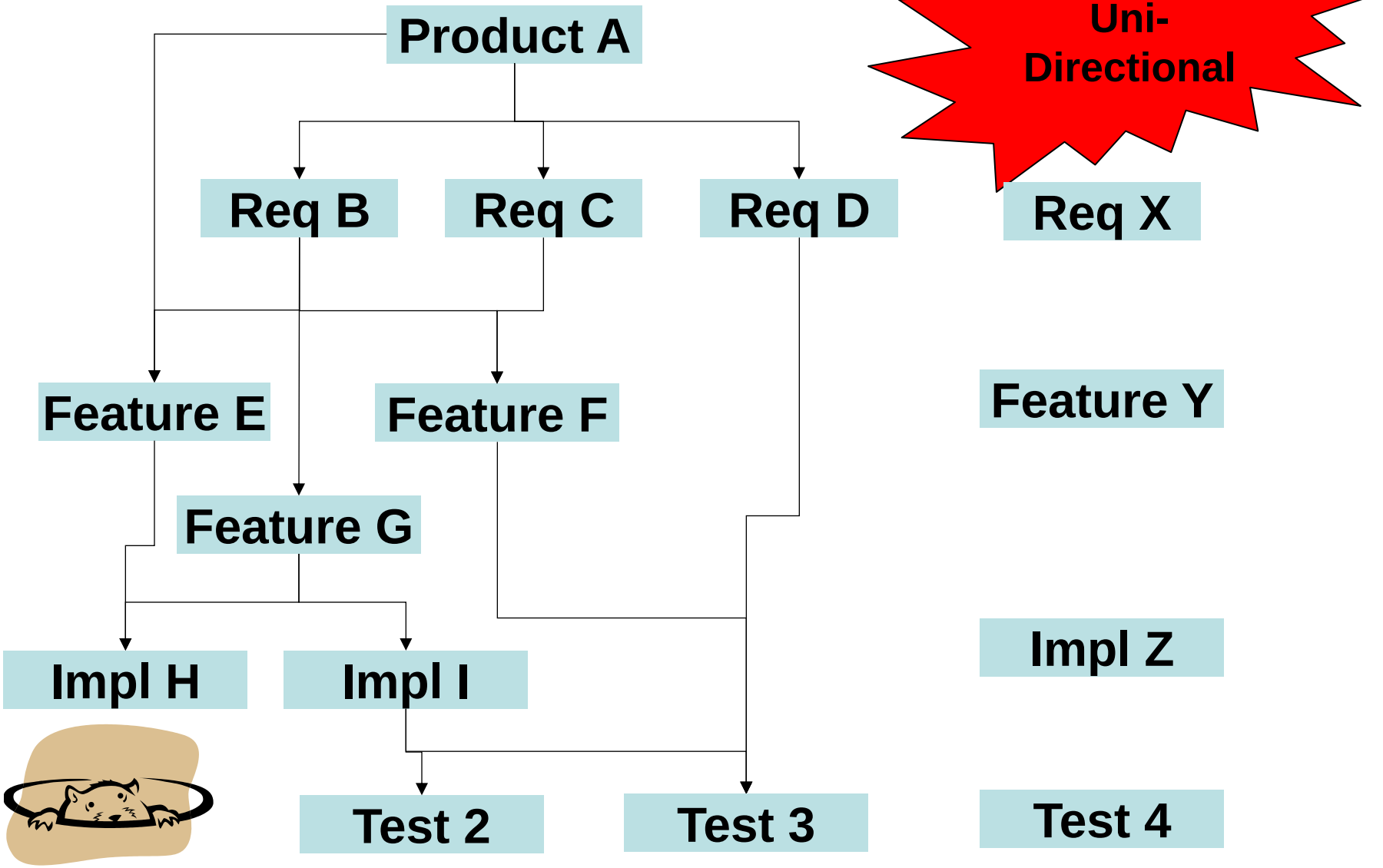
Ensuring Requirements are Implemented and Tested



Beyond tests!



Test Holes and Test Orphans!



Test Orphans and Test Holes

- **Test Orphans Waste Time and Effort!**
  - How many tests do you have where you cannot remember what they are testing?
  - What % of your test suite is like this?
  - How much time do you spend running tests where you are not sure of the value of the test?

- **Test Holes Introduce Risk!**
  - A requirement is missing a test
  - Do you know how many of your requirements are not tested?

Mapping requirements to tests

- **Many use a traceability matrix**
  - Which supports requirements tracing
  - But will not support status, results and history

Requirement Identifiers	Reqs Tested	REQ 1.1	REQ 1.2	REQ 1.3
Test Cases	**5**	**3**	**2**	**3**
1.1.1	**1**	**x**		
1.1.2	**2**		**x**	**x**
1.1.3	**2**	**x**		**x**
1.1.4	**1**			**x**
1.1.5	**2**	**x**	**x**	

Requirements management has good tool support

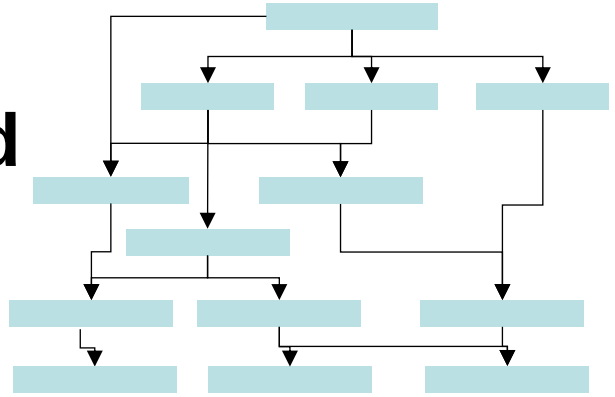
- **Example tools:**
  - Doors, Reqtify, Enterprise Architect, Jira, …
- **Requirements get mapped down to features, design, units and can even get to code**

- **Until it comes to testing**
  - At best they just map to tests without any connection to
    - Test status
    - Test results
    - Results history

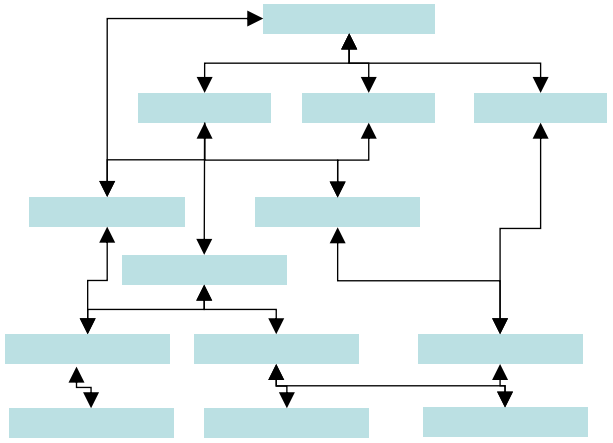
Making it difficult to track progress

Requirements tracing

- **“the ability to follow the life of a requirement, in both a backward and forward direction” [Gotel and Finkelstein, 2006]**



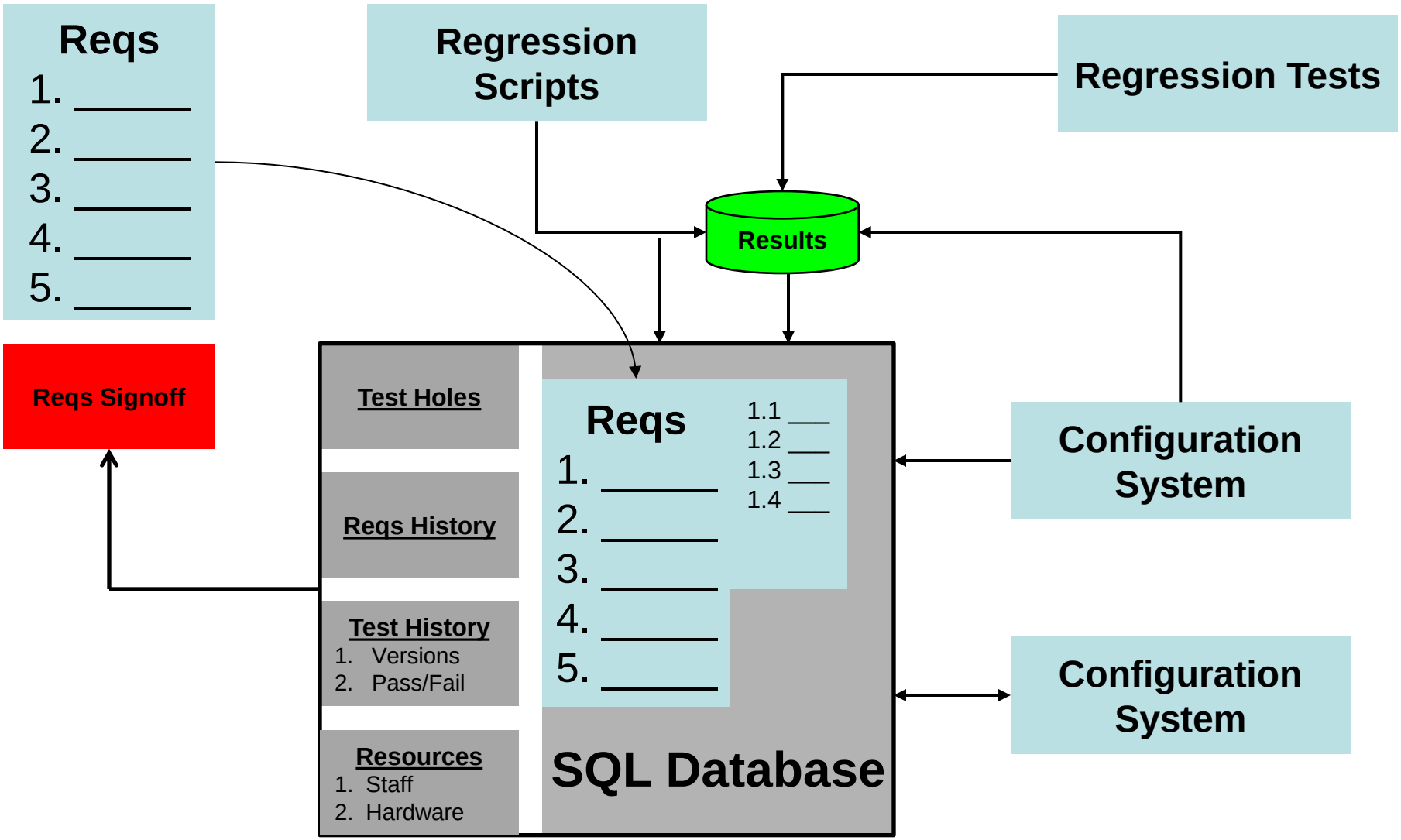
- **Requires bi-directional relationships in the requirements tree**
- **Advantages**
  - Orphan features/code
  - Business advantages – later!



Recap

- **Requirements management helps to record requirements and manage their implementation**
  - Can identify test holes
- **Bidirectional requirements mapping allows us to trace in both directions**
  - Identify orphan code and tests
  - And we will see impact and risk analysis
- **Want to also associate test status to requirements**

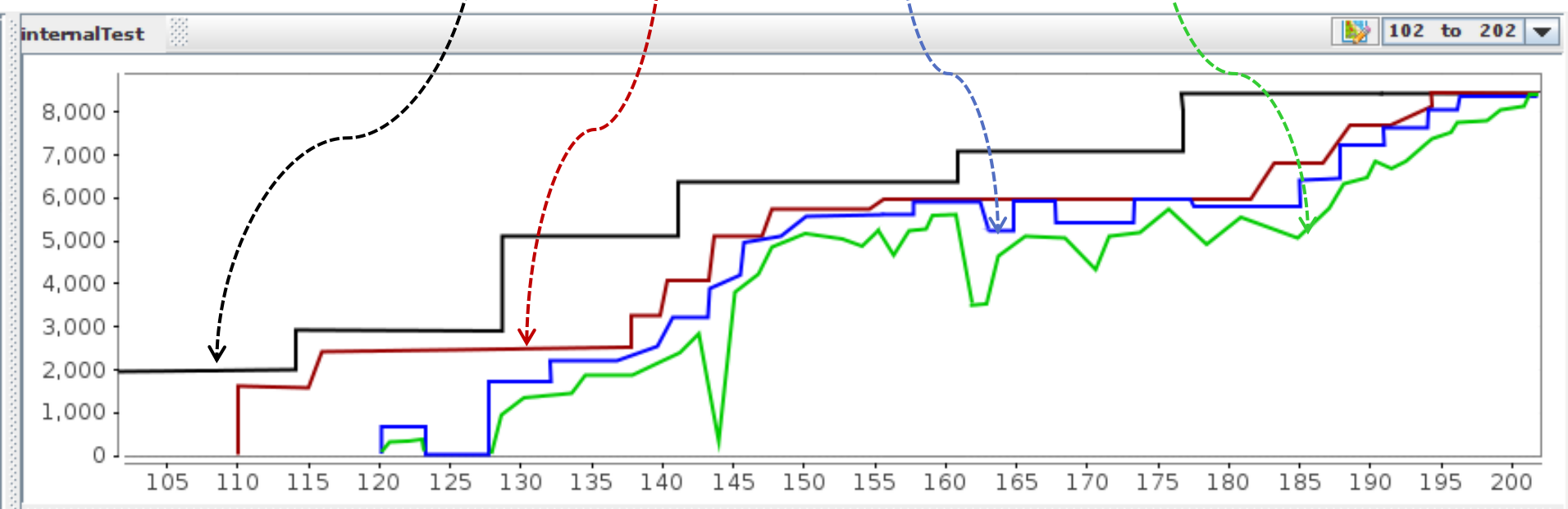
Using SQL for Requirements Management and Testing



So what do we want to track regarding testing?

Start tracking testing from the start

- Are tests defined, written, executing, passing?

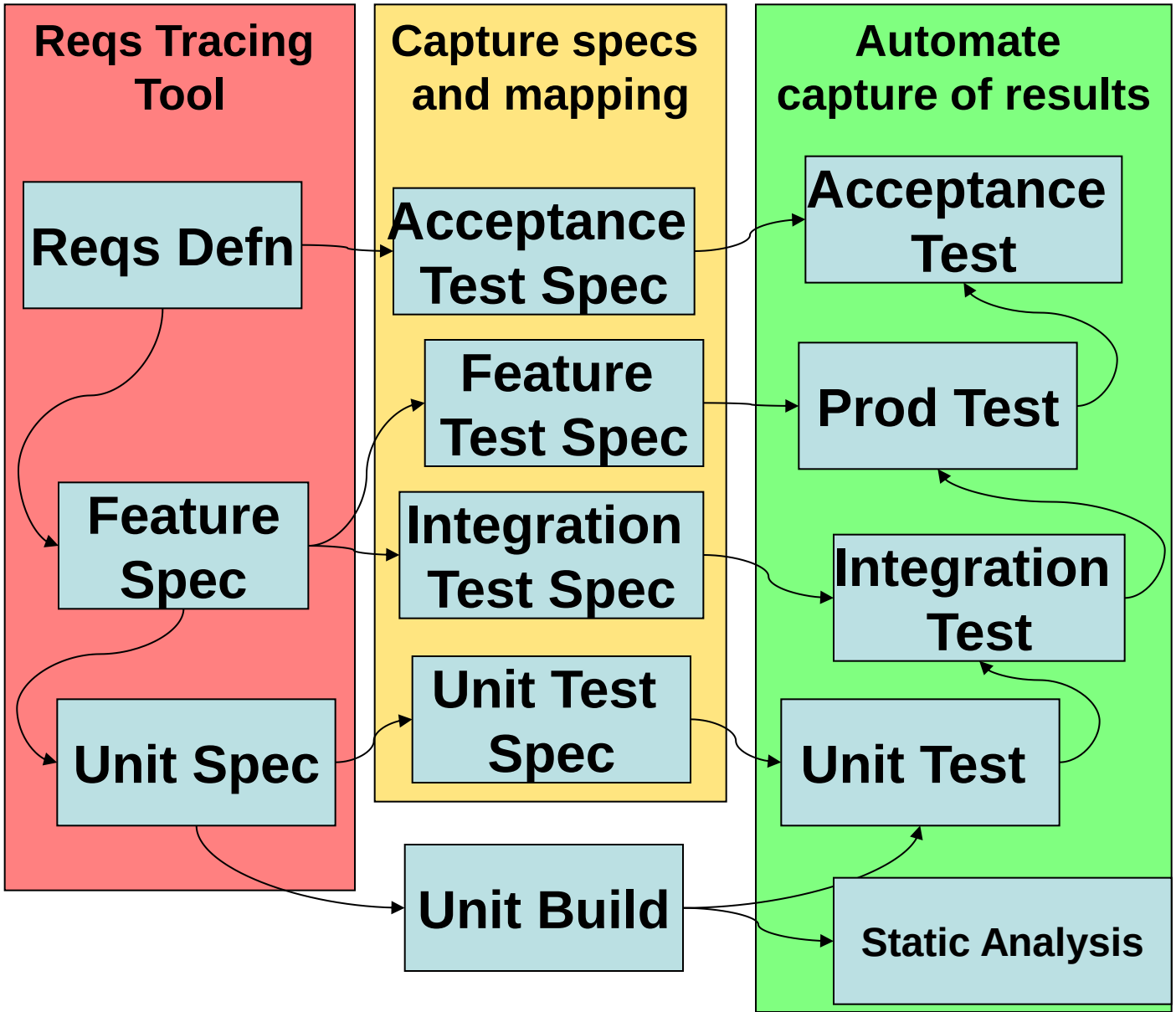


Regression	Date	Defined	Implemented	Executed	Passing
ANOTHER_VERSION_TAG_102	2011-08-20 03:13:56.0	5343	4808	4808	4738
version-103	2011-08-20 08:47:31.0	5426	4883	4883	3700
v104	2011-08-20 14:20:58.0	5503	4952	4952	4406
ANOTHER_VERSION_TAG_105	2011-08-20 19:54:24.0	5601	5040	5040	4943
a_more_verbose_description 106	2011-08-21 01:27:55.0	5639	5075	5075	5003
version-107	2011-08-21 07:01:27.0	5718	5146	4594	4529

Experiences and Recommendations

- **Importing requirements into the SQL database**
  - Using XML
  - Able to export back using XML
- **Add a simple API to get test results into the database**
  - Regression started, configuration information
  - Test started, test status
  - Regression complete
- **Extract coverage information automatically**
  - And store automatically into the database

Experiences of applying Requirements-Based Testing Supporting Sequential Development



Parallel development of verif plans and verification (TTM)

Improve specifications through test definitions

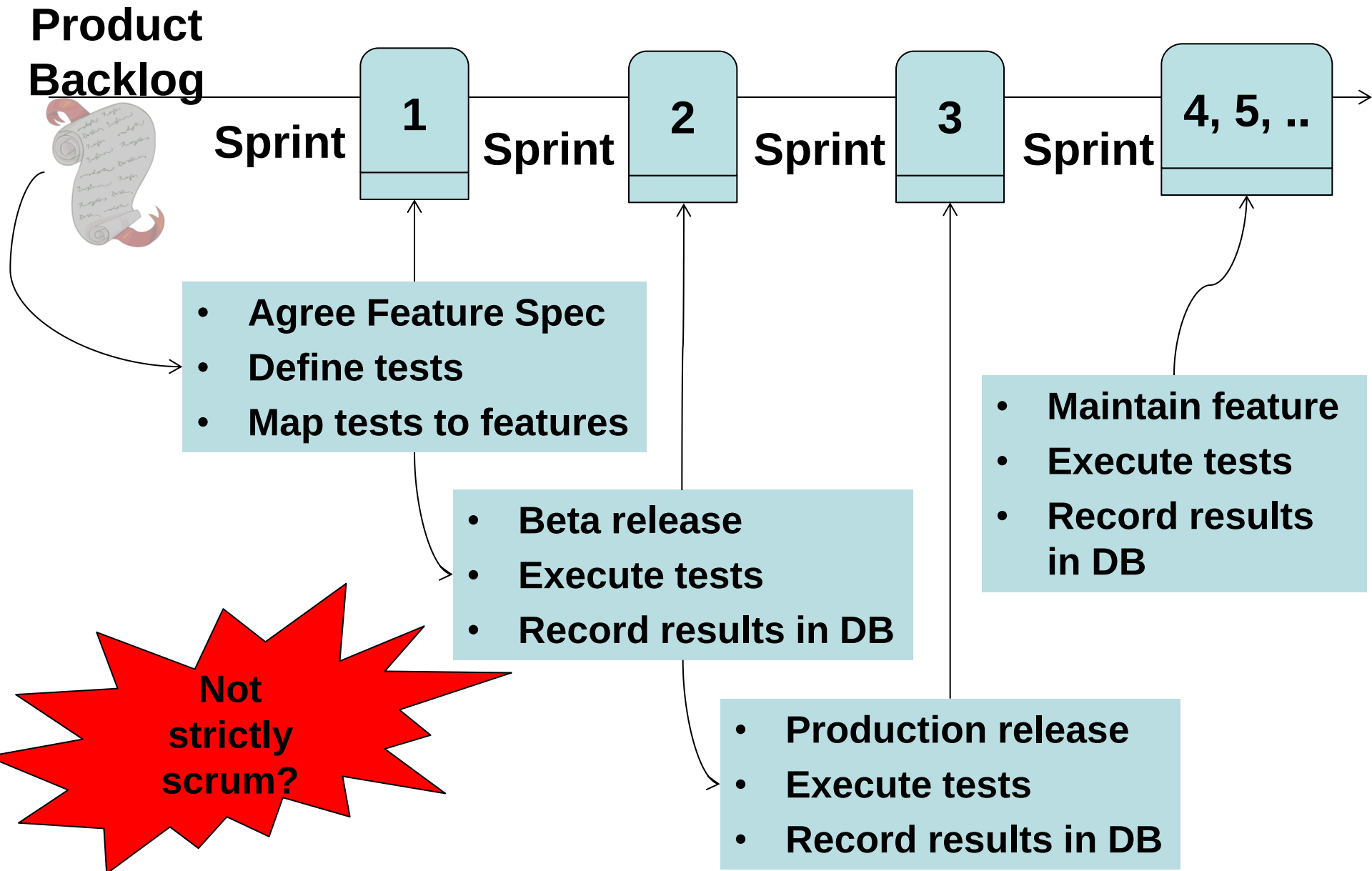
Map tests to reqs, features, etc,

Capture results

Experiences of applying Requirements-Based Testing Supporting Sequential Development

- **Important in building rigorous software systems**
  - U.S. Food and Drug Administration (FDA)
  - Mandatory for CMMI level 2 and above,
  - Mandatory for Certification in Aeronautics (DO-178B, DO-254), Railway Transportation (EN-5012x), Automotive (ISO26262,), Medical Systems (FDA 21 CFR), Other (IEC 61508), ….

Experiences of applying Requirements-Based Testing Supporting Iterative Development

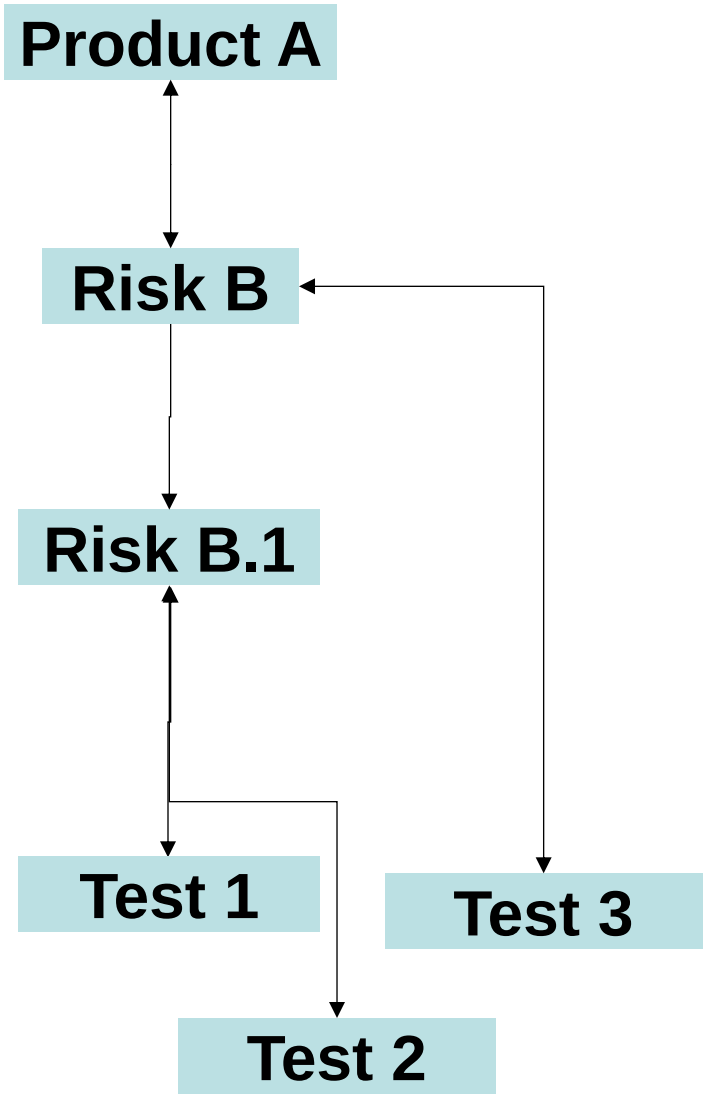


Recap

- **Requirements management helps to record requirements and manage their implementation**
- **Bidirectional requirements mapping allows us to trace in both directions**
  - Identify orphan code and tests
- **Record multiple test status rather than just pass/fail**
  - defined, written, executing, passing
- **Using an SQL database to record test data**
  - The mapping
  - The status

We will now see how this can create significant business advantage

Risk-based Testing

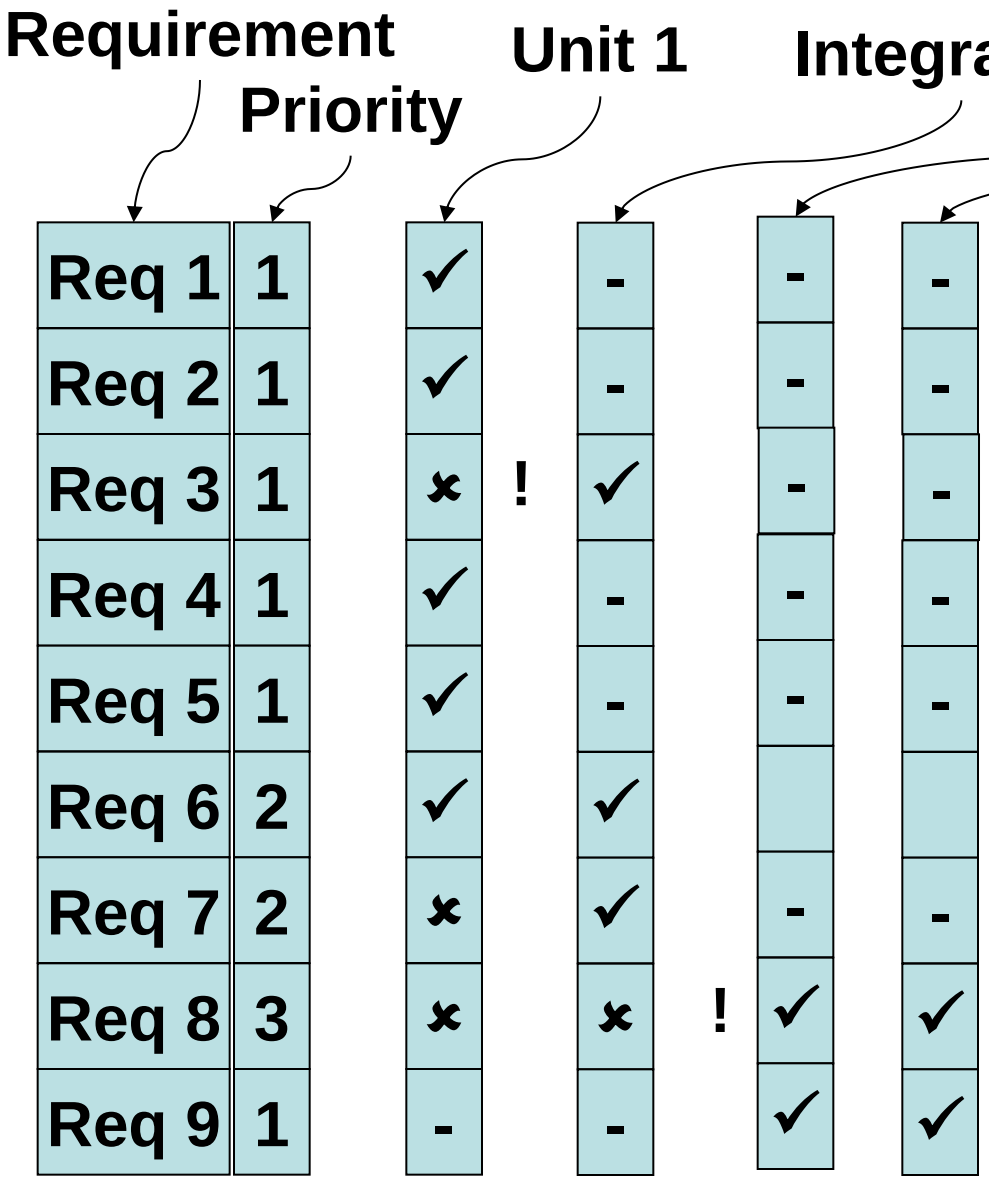


- **Risk**
  - Search for clients whose total value of sales between 2 specified dates is above a specified value returns wrong result
  - Probability = Low
  - Impact = High

Risk

- **Tests**
  - Query test database XXX with dates “d1/m1/y1” and “d2/m2/y2”. Expected result = “client1”,...
- **Searching the test database**
  - Prioritise tests according to risk
  - Calculate remaining risk
    - A passing test mitigates risk

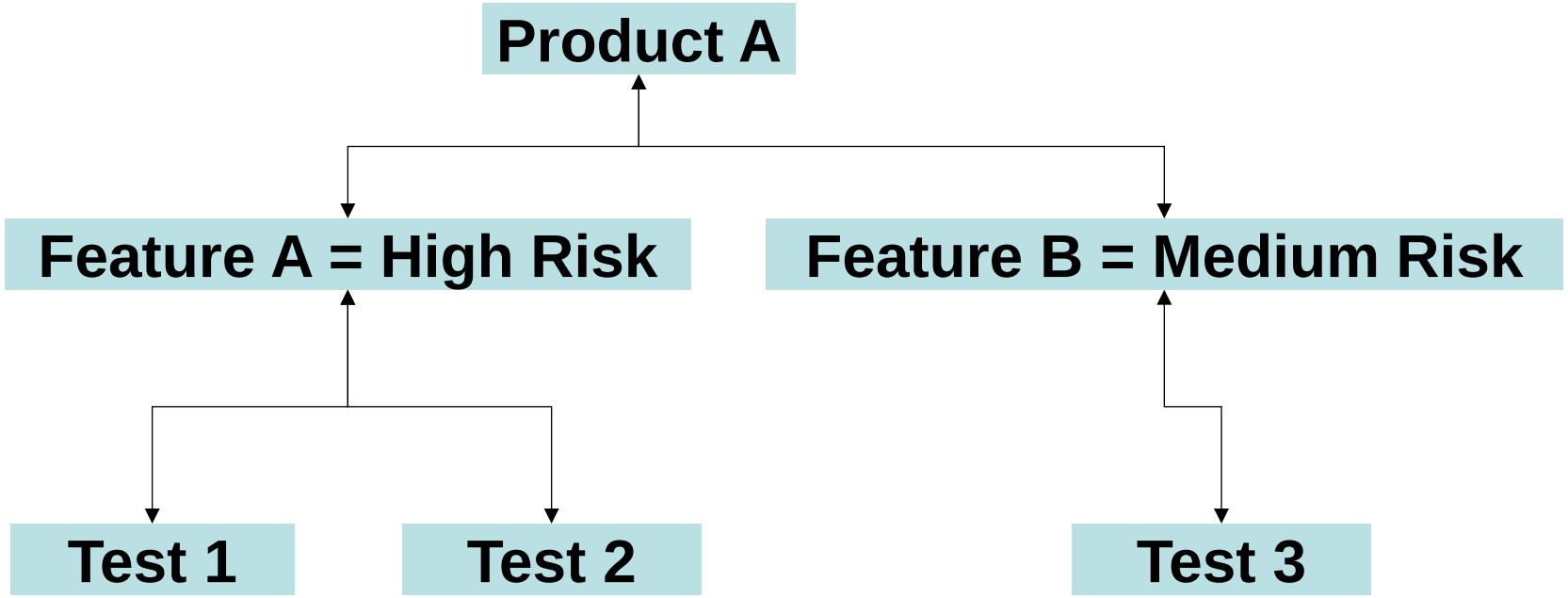
Improved Time-To-Market through Prioritisation and Risk Analysis



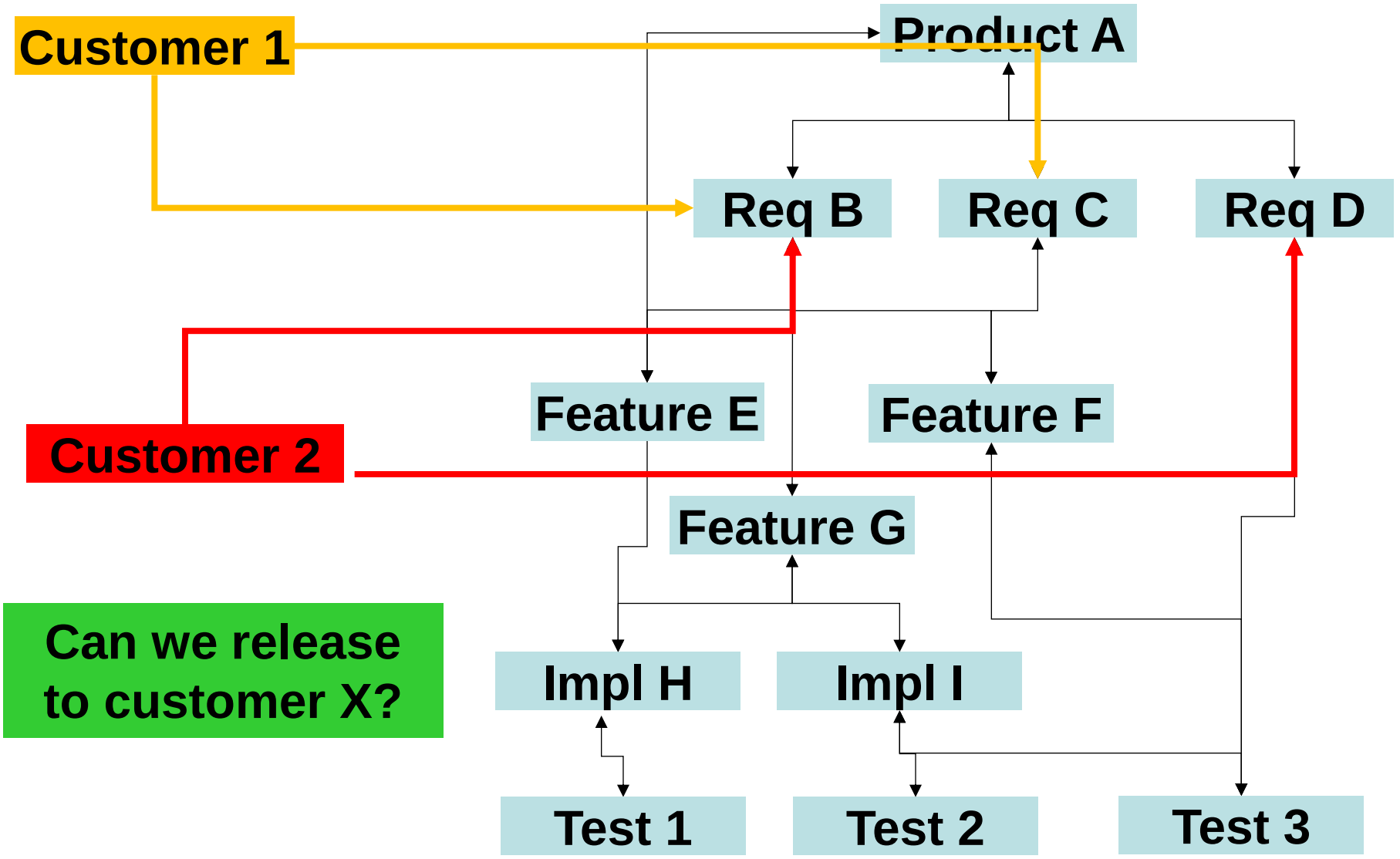
- **Release Unit level with known risk**
  - Close at higher level
- **Release with known risk**

Another view of Risk-based Testing (product risk)

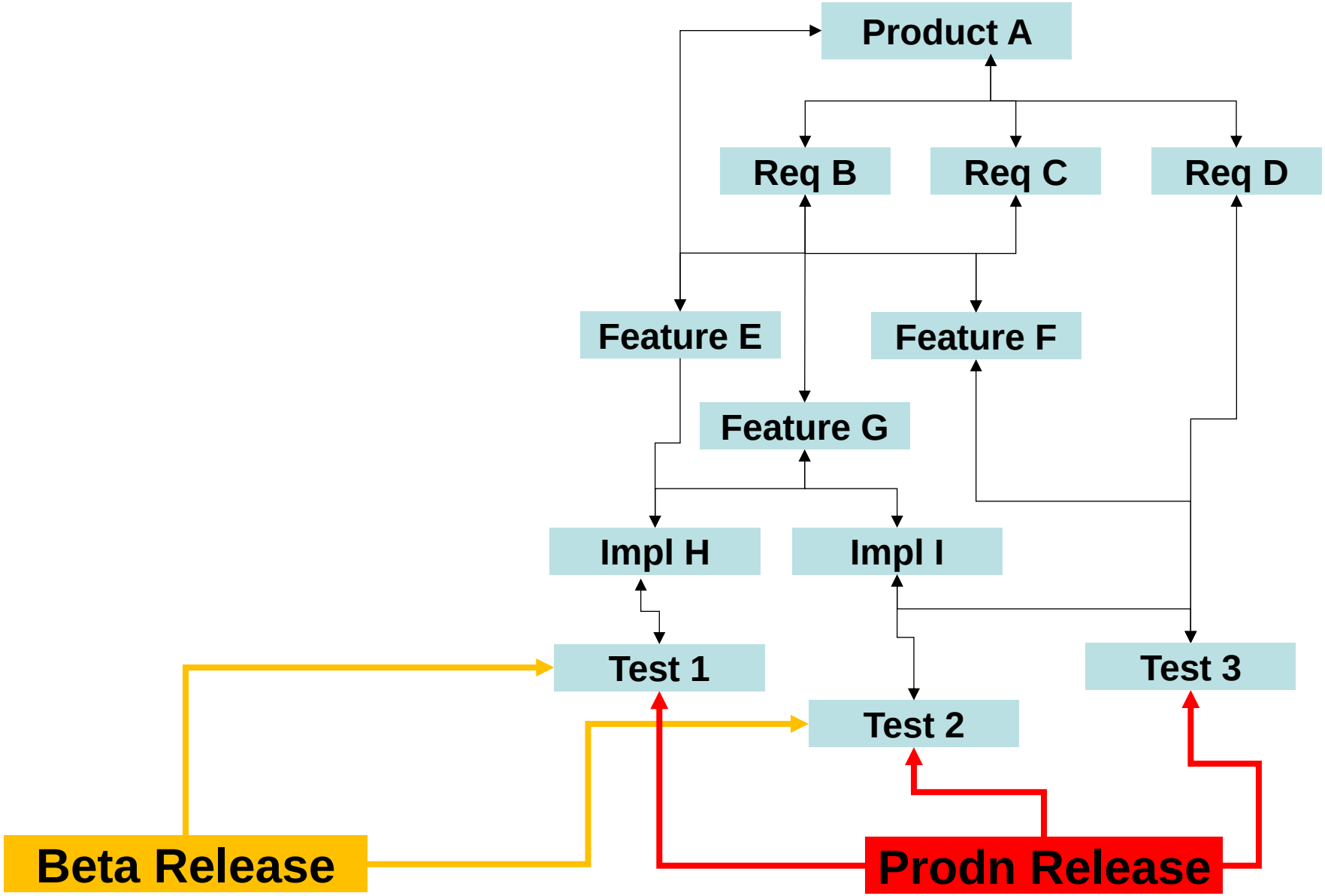
- **Higher Risk = More testing**
  - Higher risks should have more tests
  - Can add “risk field” to features based on likelyhood of failure and impact of failure
  - Can match level of testing to risk and ensure sufficient level of testing (using easy searches)



Filtering Requirements based on Customers



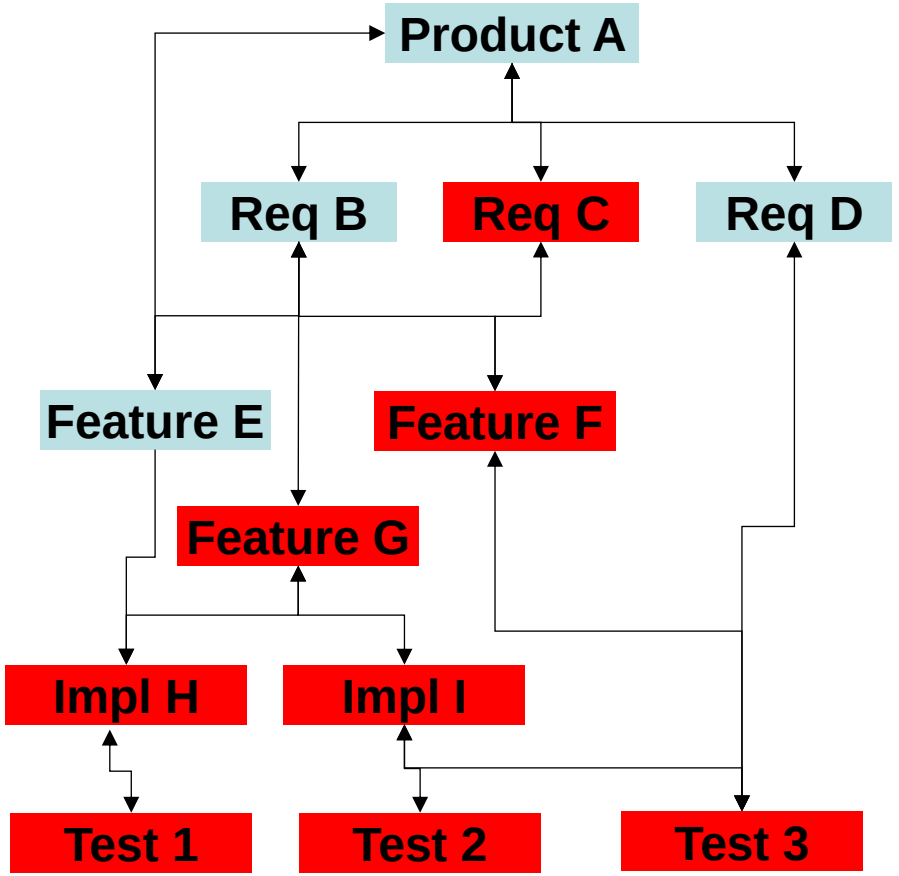
Filtering Based on release



Impact Assessment

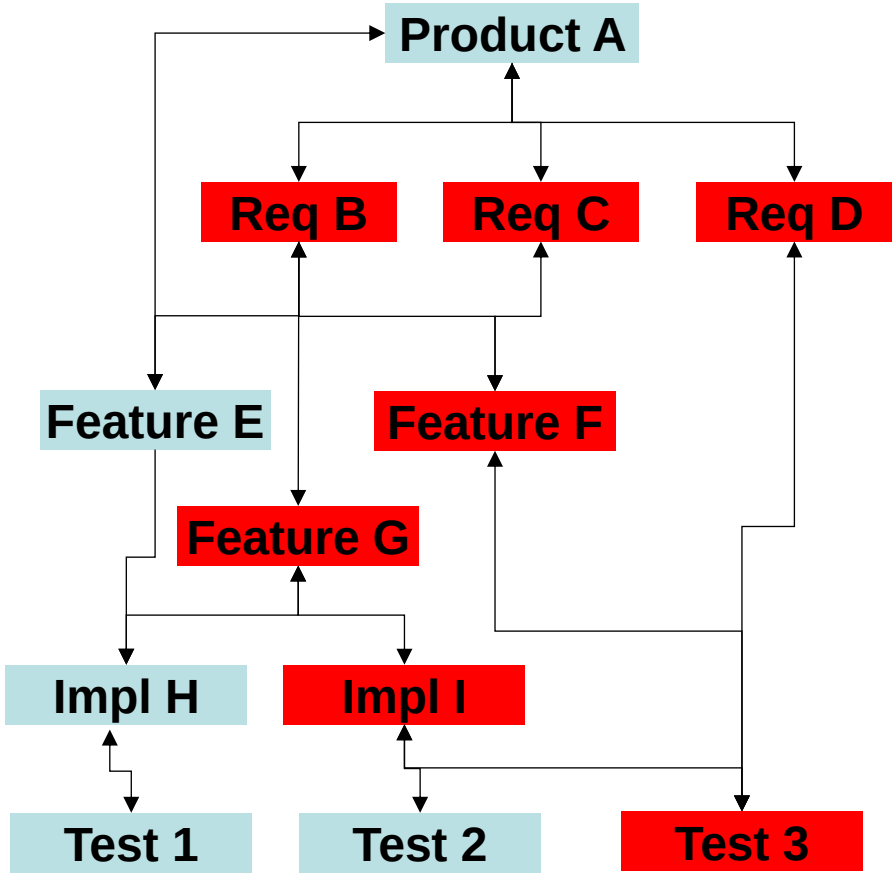
- How often do people ignore testing when assessing the impact of a change?

- What is the impact on changing Req C?



Impact Assessment

- **What is the impact on dropping test 3?**



Return on Investment?

- **Initial investment is relatively high**
  - Building initial SQL database
  - Ensuring requirements are recorded
  - Ensuring test information is stored
  - Mapping requirements to tests
- **Business advantage is huge**
  - Identify test holes and test orphans
  - Understanding status at all points in project
    - “Defined, written, running, passing”
  - Automation of analysis (risks, impact, release readiness)
  - Trend analysis
    - Better prediction of release readiness

Summary

- **Map requirements to tests**
- **Database to record mappings and results**
  - Store results from test automation
  - Record %'s of tests defined, written, executing, passing
- **Advantages**
  - Identify test holes and test orphans
  - Track the status of the whole verification effort
  - Use historical perspective for more accurate predictions
  - Better reporting of requirements status
  - Support for
    - Risk-based testing
    - Prioritisation and Risk Analysis
    - Filtering Requirements based on Customers and releases
    - Impact analysis
  - Support for regulatory-based requirements signoff

Q&A

- **Mike Bartley**
  - mike@testandverification.com
  - Mike Bartley on LinkedIn
  - Other materials on www.testandverification.com