SIEMENS
Ingenuity for life

PLM and Innovation Excellence

Learning Campus

Your partner for Business Learning

Siemens Core Learning Program

Test Strategies

Author:
Peter Zimmerer, CT

… hit the market window … like a bird …

Test strategies

Learning objectives

- Understand what a test policy and test strategy is all about
- Get to know different kinds of test strategies
- Learn how to create and drive a risk-based test strategy
- Get to know important test levels and types

Test Strategies

Agenda

Strategies for Testing

Risk-Based Testing

Test Levels & Types

Summary

Test policy

*Testing is a journey that serves a purpose (**test mission**), that must have objectives and a destination (**test policy**), and that requires a map (**test strategy**)*

SIEMENS
Ingenuity for life

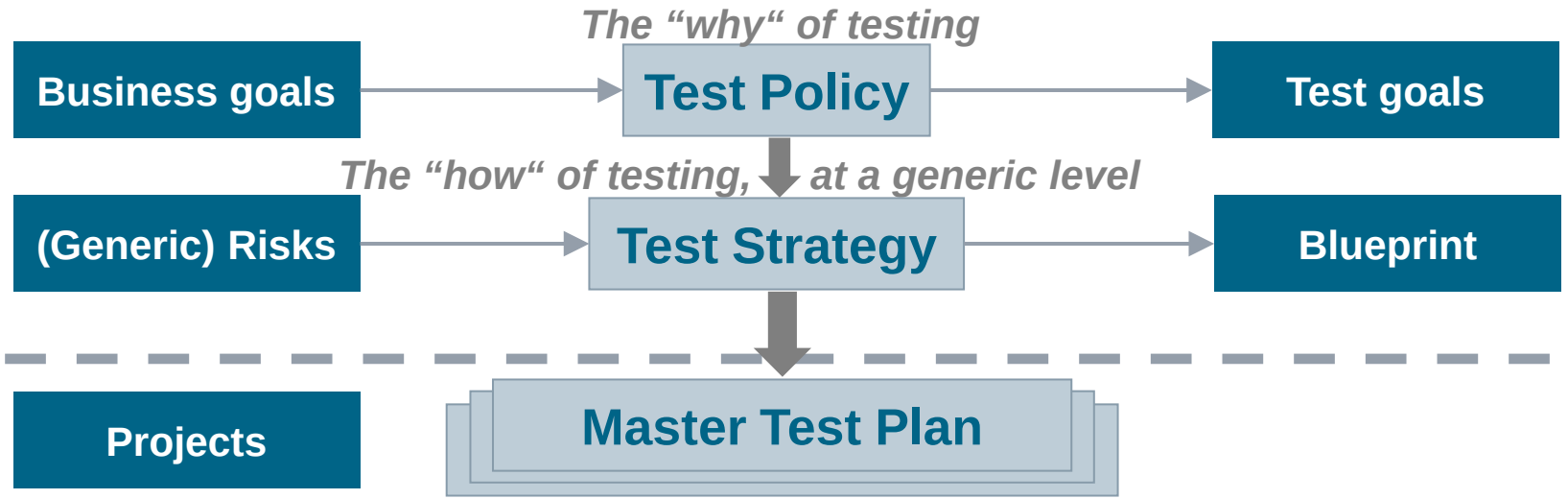
ISTQB Glossary (http://www.istqb.org/)

Test mission – The purpose of testing for an organization, often documented as part of the test policy.

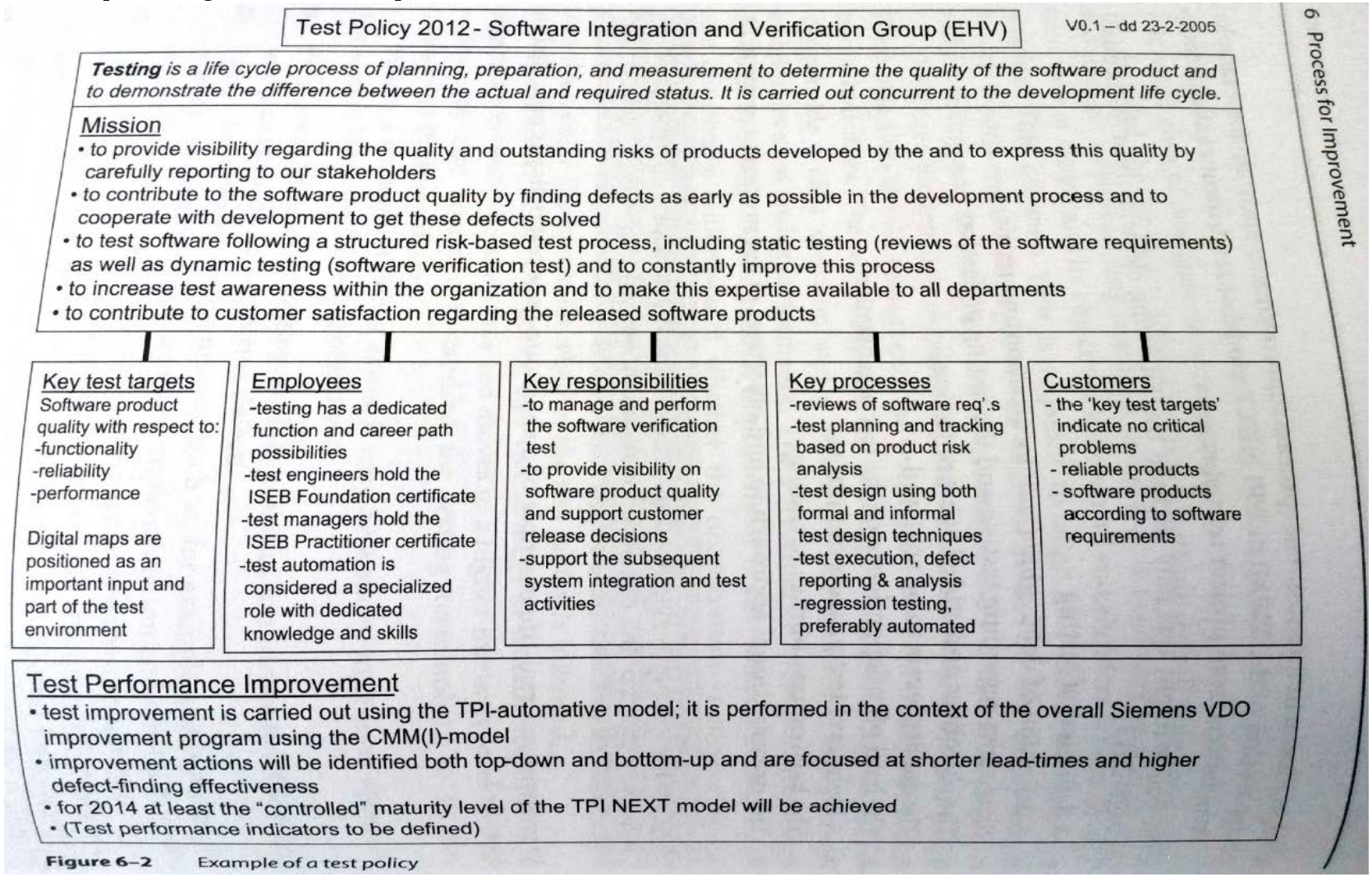
Test policy – A high level document describing the principles, approach and major objectives of the organization regarding testing.

ISO/IEC/IEEE 29119-1

Organizational Test Policy – an executive-level document that describes the purpose, goals, and overall scope of the testing within an organization, and which expresses why testing is performed and what it is expected to achieve.



Test policy – Example



Test Policy 2012 - Software Integration and Verification Group (EHV)

V0.1 – dd 23-2-2005

6 Process for Improvement

Testing *is a life cycle process of planning, preparation, and measurement to determine the quality of the software product and to demonstrate the difference between the actual and required status. It is carried out concurrent to the development life cycle.*

Mission

- *to provide visibility regarding the quality and outstanding risks of products developed by the and to express this quality by carefully reporting to our stakeholders*
- *to contribute to the software product quality by finding defects as early as possible in the development process and to cooperate with development to get these defects solved*
- *to test software following a structured risk-based test process, including static testing (reviews of the software requirements) as well as dynamic testing (software verification test) and to constantly improve this process*
- *to increase test awareness within the organization and to make this expertise available to all departments*
- *to contribute to customer satisfaction regarding the released software products*

Key test targets

Software product quality with respect to:
-functionality
-reliability
-performance

Digital maps are positioned as an important input and part of the test environment

Employees

-testing has a dedicated function and career path possibilities
-test engineers hold the ISEB Foundation certificate
-test managers hold the ISEB Practitioner certificate
-test automation is considered a specialized role with dedicated knowledge and skills

Key responsibilities

-to manage and perform the software verification test
-to provide visibility on software product quality and support customer release decisions
-support the subsequent system integration and test activities

Key processes

-reviews of software req'.s
-test planning and tracking based on product risk analysis
-test design using both formal and informal test design techniques
-test execution, defect reporting & analysis
-regression testing, preferably automated

Customers

- the 'key test targets' indicate no critical problems
- reliable products
- software products according to software requirements

Test Performance Improvement

- test improvement is carried out using the TPI-automative model; it is performed in the context of the overall Siemens VDO improvement program using the CMM(I)-model
- improvement actions will be identified both top-down and bottom-up and are focused at shorter lead-times and higher defect-finding effectiveness
- for 2014 at least the "controlled" maturity level of the TPI NEXT model will be achieved
- (Test performance indicators to be defined)

Figure 6–2 Example of a test policy

Reference: Graham Bath und Erik van Veenendaal: Improving the Test Process

Test strategy (1)

ISTQB Glossary (http://www.istqb.org/)
Test strategy – A high-level description of the test levels to be performed and the testing within those levels for an organization or programme (one or more projects).

ISO/IEC/IEEE 29119-1
Test Plan – Detailed description of test objectives to be achieved and the means and schedule for achieving them, organised to coordinate testing activities for some test item or set of test items.
Test strategy – Part of the **Test Plan** that describes the approach to testing for a specific test project or test sub-process or sub-processes.

Test Strategy 4
Critical Parts

TMap® (Test Management Approach by Sogeti)
Test strategy – The distribution of the test effort and coverage over the parts to be tested or aspects of the test object aimed at finding the most important defects as early and cheaply as possible.

Test strategy (2)

iX Studie Software-Testmanagement (2006)
Teststrategie – Die Verknüpfung der zur Prüfung der Testobjekte einzusetzenden Testverfahren, den Ansatz zur Verteilung der zur Verfügung stehenden Ressourcen auf die zu testenden Teile und die Definition der Reihenfolge der durchzuführenden Aktivitäten bezeichnet man als Teststrategie.

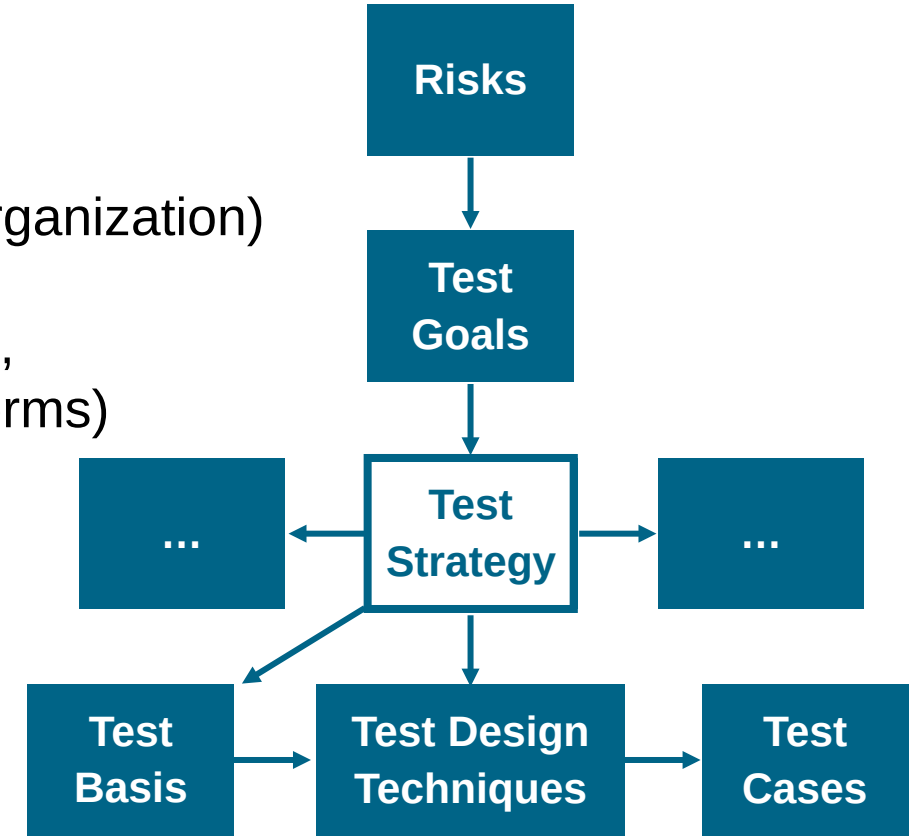
(Test strategy – The mapping of used testing techniques to check objects under test, the approach of distribution of available resources on parts to be tested and the definition of order of tasks).

Typically the test strategy is documented in a Master Test Plan
But a **good test strategy** is not “just another document”
but a **real strategy** explaining stakeholders the rationale (why?),
what you are trying to achieve, and how you are going to do it

Test strategy (3)

Ingredients of a test strategy
Why? What to test? Where? Who? When? How (much)?

- Risks, test goals
- Test object, system under test
- Test effort distribution
- Test process (roles, responsibilities, organization)
- Test management (testware artifacts)
- Test basis (specs, documents, models, optional / mandatory standards and norms)
- Test levels
- Test methods, test design techniques
- Entry and exit criteria (measures)
- Test automation (architecture)
- Regression testing
- Defect management



Test strategy (4)

A Test Strategy provides rationale and decisions to link and relate from goals, objectives, values, benefits, risks to delivering on those values and addressing the major constraints, risks, priorities

Test strategy – Three basic questions

SIEMENS
Ingenuity for life

Why bother?

- Testing is expensive, so do the testing as effectively and as efficiently as possible
- Only activities addressing any risk should be included

Who cares (→ stakeholder analysis)?

- Testing is no end in itself
- Who will receive and accept the results?
- Who will pay for the results?
- Who is our sponsor?

How much?

- Exhaustive testing is not possible
- What is an adequate test set?

Test strategy – Drivers

Risks
- *No risk – No test!*

Quality criteria

Requirements
- Business goals, product mission, standards, norms, quality criteria

Development
- Project lifecycle, project management, development team
- Test entry criteria (e.g. *testability* aspects)

People (test team)

Environment (test lab)

Mission
- Minimize time and / or cost
- Find as many bugs as possible
- Find important problems only, assess perceived quality risks
- Certify to standard, verify a specification
- Focus on defect prevention and not on defect detection

Test strategies – Types (1a)

SIEMENS
Ingenuity for life

Analytical

- object-guided: use requirements, design, and implementation objects to determine testing focus
- risk-based: assess and prioritize quality risks
- fully-informed: intensive analysis of test basis, system, usage profiles

Model-based

- scenario-based: use real-world scenarios, statistical testing
- use-case-based: from the object-oriented world
- domain-based: analyze domains of input data

Methodical

- learning-based: use checklists, develop them over time
- function-based: identify and test every function
- state-based: identify and test every state
- quality-based: quality hierarchy like ISO 9126 / 25000 (-ilities)
- failure-based: use failure data and experience

Reference: Rex Black: Pragmatic Software Testing

Test strategies – Types (1b)

SIEMENS
Ingenuity for life

Process-oriented

- standardized: follow official or recognized standards like IEEE 829 or ISO/IEC/IEEE 29119 or develop your own process steps
- agile: lightweight processes
- automation: automated random, automated functional

Dynamic

- intuitive: collective experience, wisdom, and instincts
- exploratory: simultaneous test design, test execution, and learning
- bug hunting: bug profiles, taxonomies, bug assumptions

Philosophical

- exhaustive
- shotgun: randomly distribute the test effort like pellets from a shotgun
- externally-guided

Regression and regression risk

- repeat all tests: brute force, requires automation in place
- repeat some tests: traceability, change analysis, quality risk analysis

Reference: Rex Black: Pragmatic Software Testing

Test strategies – Types (2)

Analytical (analyze the test basis)
- requirements-based testing
- risk-based testing

Model-based (derive testware from models)
- operational profiling (e.g. reliability and performance testing), behavior models

Methodical
- quality characteristic-based, quality standards, checklists

Process- or standard-compliant
- predefined processes or standards

Reactive
- defect-based attacks
- exploratory testing, heuristics

Consultative (rely on input of key stakeholders)
- user-directed testing (e.g. outsourced compatibility testing)

Regression-averse
- extensive test automation, reusable testware

Reference: ISTQB Test Manager Syllabus (Advanced Level, Expert Level), ISTQB Glossary

Test Strategies

Agenda

Strategies for Testing

Risk-Based Testing

Test Levels & Types

Summary

What is a risk?

A risk is an unwanted event that threatens (project) objectives with negative consequences

Three aspects are related to risks

- likelihood of occurrence
- impact (loss, cost)
- degree to which its outcome can be influenced

There are different categories of risks

- Project risks: hard deadlines, external dependencies, missing skills
- Process risks: planning risks, underestimation, bad progress control
- ***Product risks***: bad / unstable requirements, bad usability, product complexity, fault proneness, bad quality / stability / performance

Testers (or a cross-functional risk team) own the *risk process*, but not the risks!

Test strategy – Based on risks

SIEMENS
Ingenuity for life

Base the testing on business goals, values, priorities, constraints
→ *Risk-based test strategy (RBT)*

Continual releases, limited resources
Triage of new features vs. technical debt
Ship early with confidence

No risk – No test
Project risks – Process risks – *Product risks* (focus in testing)
→ Product risk analysis

A test strategy should answer:
- **Why?**
- **What to test? Where?**
- **Who tests? When to test?**
- **How to test?**
- **How much to test?**

Risk = P × D

P Probability of failure
- Frequency of use
- Chance of failure:
  criticality & complexity at implementation & usage, lack of quality

D Damage (consequence & cost for business & test & usage)

Risk analysis – General sources (1)

SIEMENS
Ingenuity for life

Inside-Out

- *What risks are associated with this thing?*
- *What can go wrong here?*
- Vulnerabilities
  - What weaknesses or possible failures are there in this component?
- Threats
  - What inputs or situations could there be that might exploit a vulnerability and trigger a failure in this component?
- Victims
  - Who or what would be impacted by potential failures and how bad would that be?

Risk analysis – General sources (2)

SIEMENS
Ingenuity for life

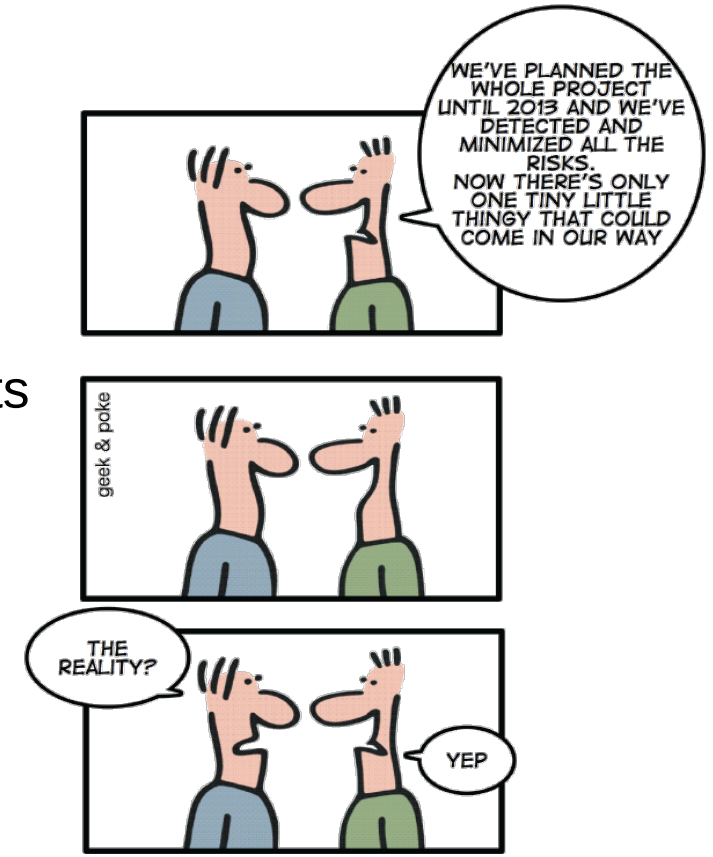
Outside-In

- *What things are associated with this kind of risk?*
- Predefined set of potential risks, specific risk taxonomy
- Quality criteria categories (like ISO 9126, ISO 25000)
  - Accessibility, capability, compatibility, concurrency, conformance to standards
  - Efficiency, installability and uninstallability, localizability, maintainability
  - Performance, portability, recoverability, reliability, scalability, security
  - Supportability, testability, usability
- Generic risk lists, risk catalogs, error taxonomies as problem drivers
  - Complex, new, changed
  - Dependencies, distribution
  - Critical, precise, popular, strategic
  - Third-party, buggy, recent failure
  - Specific for application domain, e.g. client / server, eBusiness, mobile, etc.
- Use (post-mortem) experiences and fault analysis from last projects

Risk analysis – General sources (3)

Do not forget *generic test objectives*, e.g.:

- Check components meet requirements
- Check components are ready for integration
- Check system meets functional requirements
- Check system meets nonfunctional requirements



Do not forget general *testing wisdom*

- Use best practices of testing
  - E.g. mandatory testing activities for C++:
    - Static testing of source code
    - Checking of memory management
- Use know-how and experience of testing experts
- Use your own experience from your specific domain

Example:
Product risk analysis for software architects

SIEMENS
Ingenuity for life

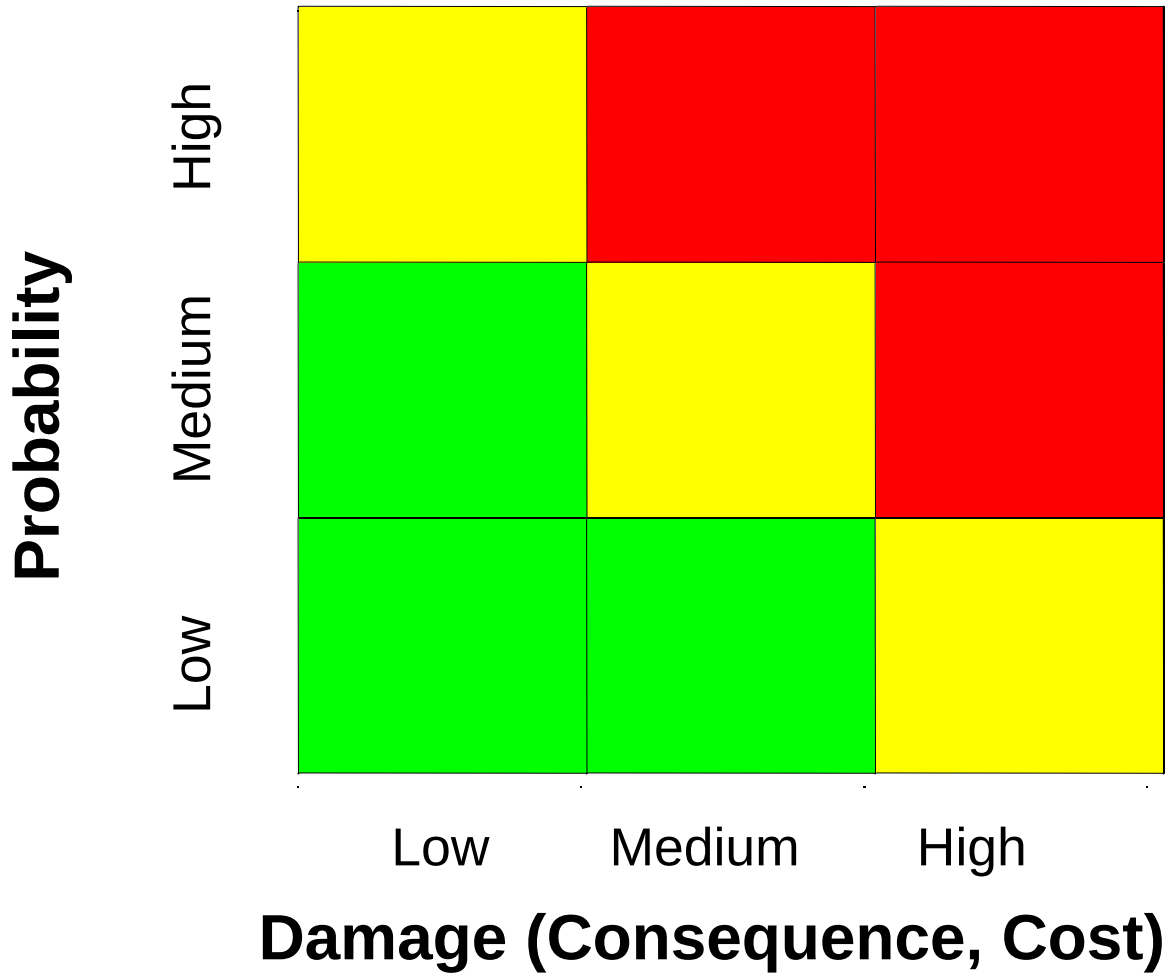
Identify especially all risks related to the architecture

- Architectural requirements (functional and non-functional)
- Architectural quality attributes, claims
- Architectural use cases, features, functions, epics, user stories, processes, services, APIs
- Architectural decisions
- Design decisions
- Technology selections
- Third-party component selections: frameworks, tools, open source, external partnering
- Core assets in domain engineering of product line engineering
- Open variant space in software ecosystems
- Bug categories

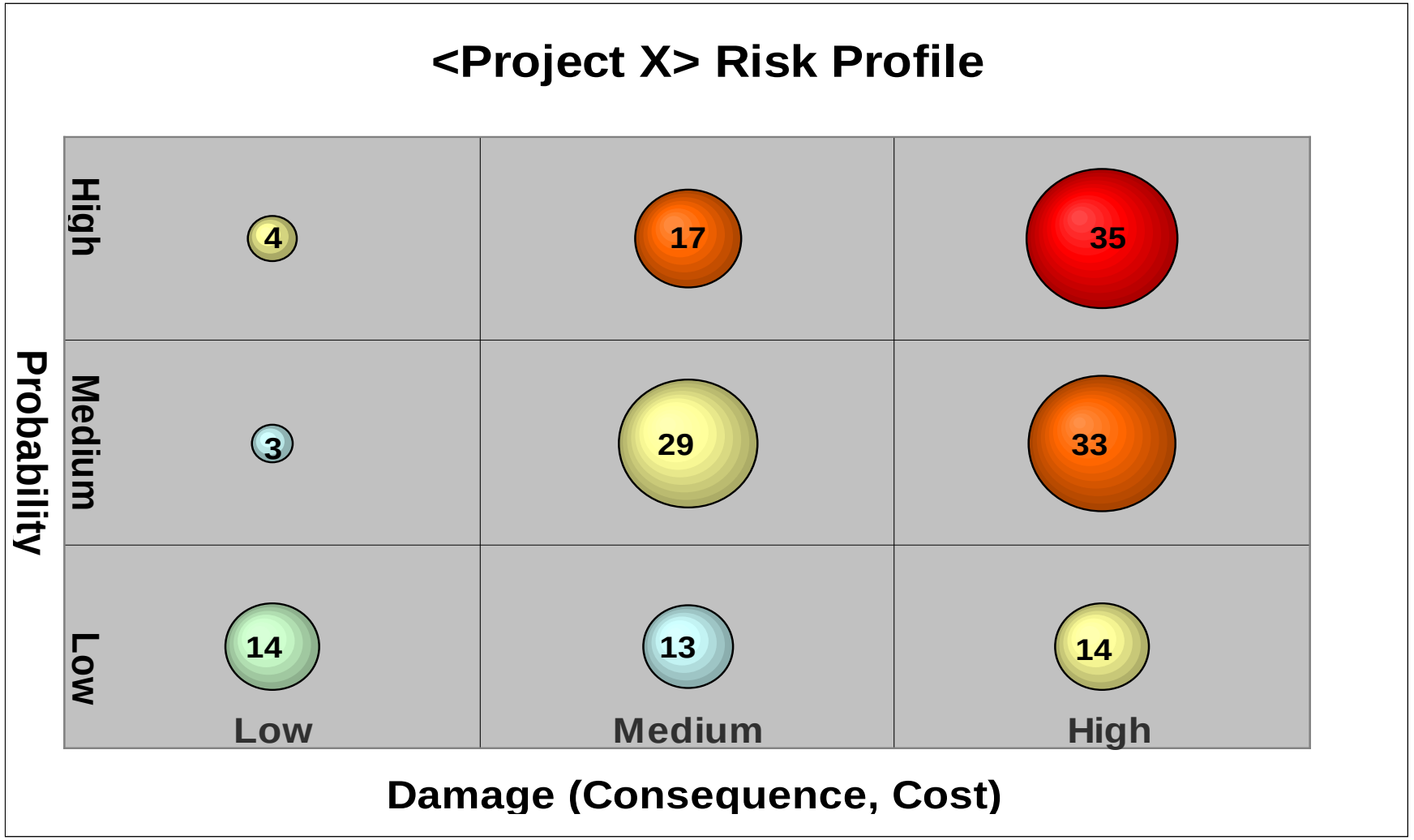
Actively participate in a product risk analysis workshop as one important stakeholder

Risk identification and analysis is the responsibility of all stakeholders; testers can facilitate and drive this process

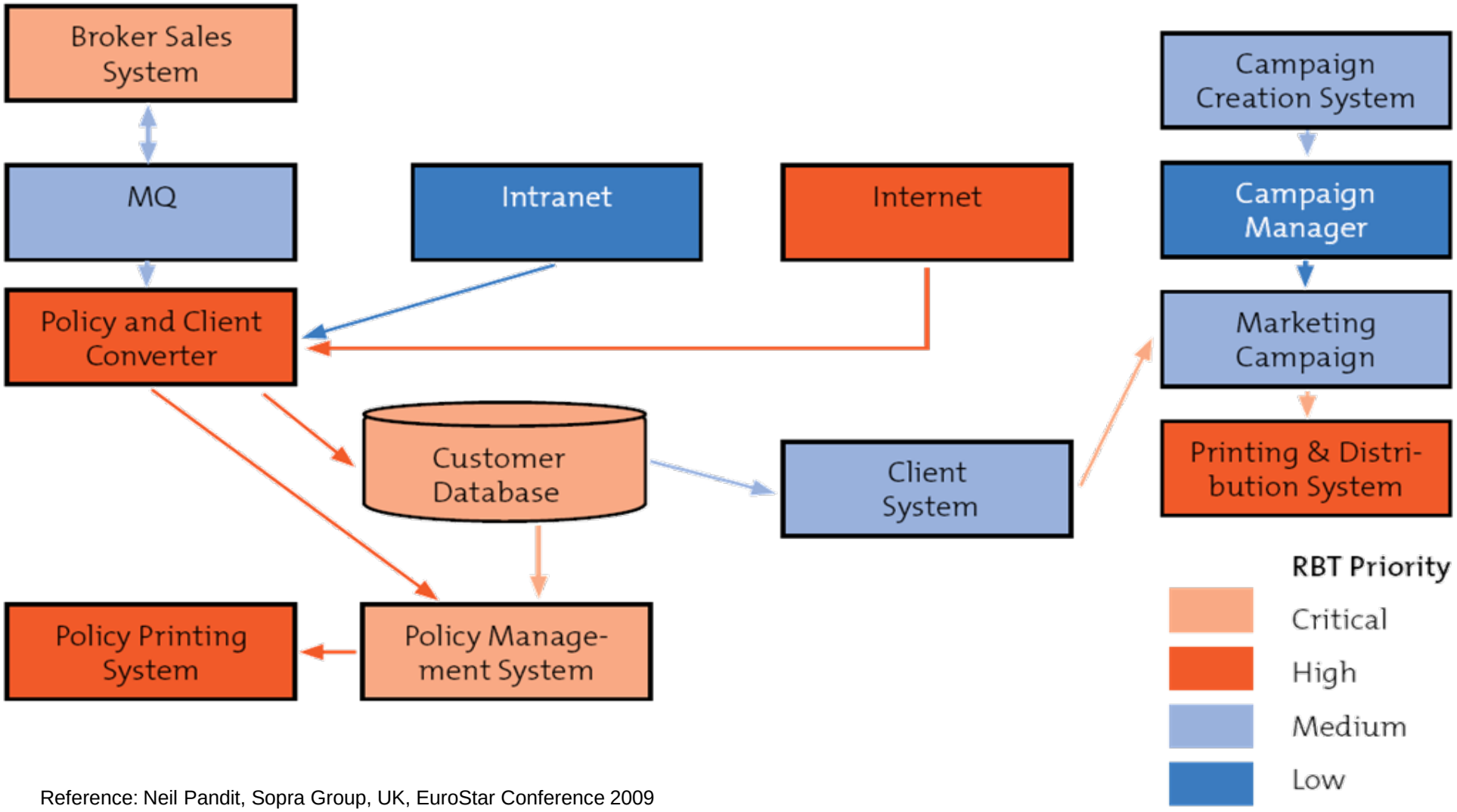
Quantification / Categorization of risks



Quantification / Categorization of risks – Example

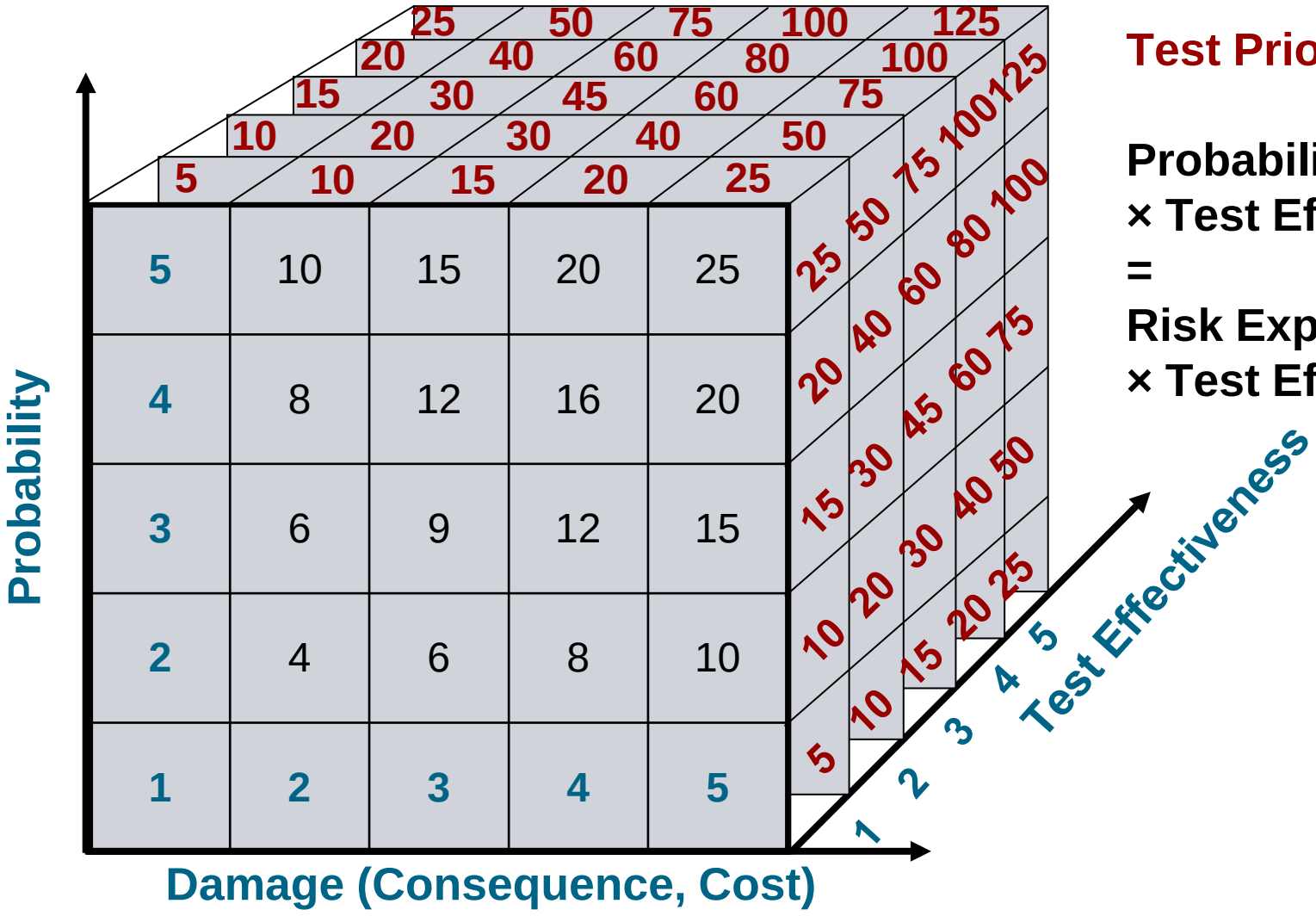


Quantification / Categorization of risks – Heatmap



Reference: Neil Pandit, Sopra Group, UK, EuroStar Conference 2009

Quantification of risks – Example



Test Priority Number

Probability × Damage × Test Effectiveness = Risk Exposure × Test Effectiveness

Risk-based test strategy (RBT)

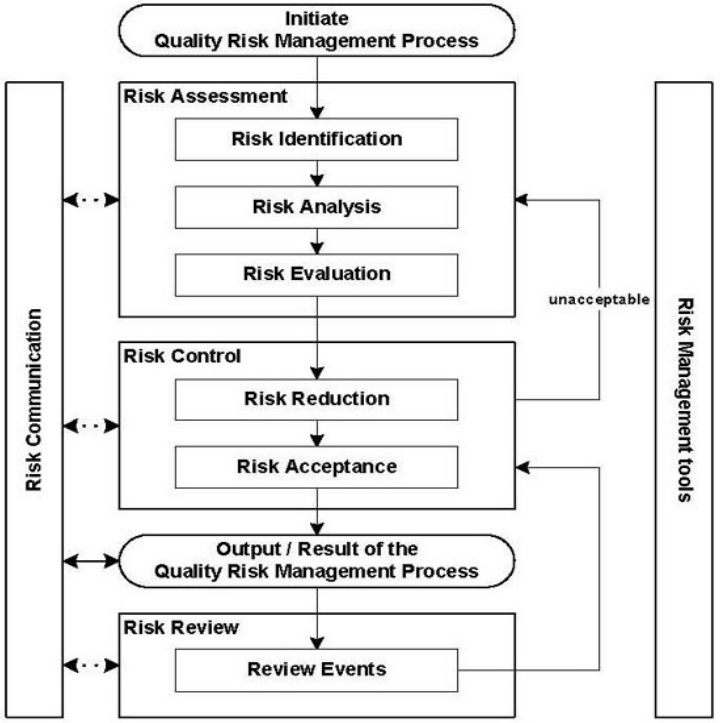
Use categorized risks to

- Work on riskiest stuff first
- Determine the extent / intensity / depth of testing to be carried out
- Focus and prioritize testing activities and test cases
- Determine the test design techniques to be used

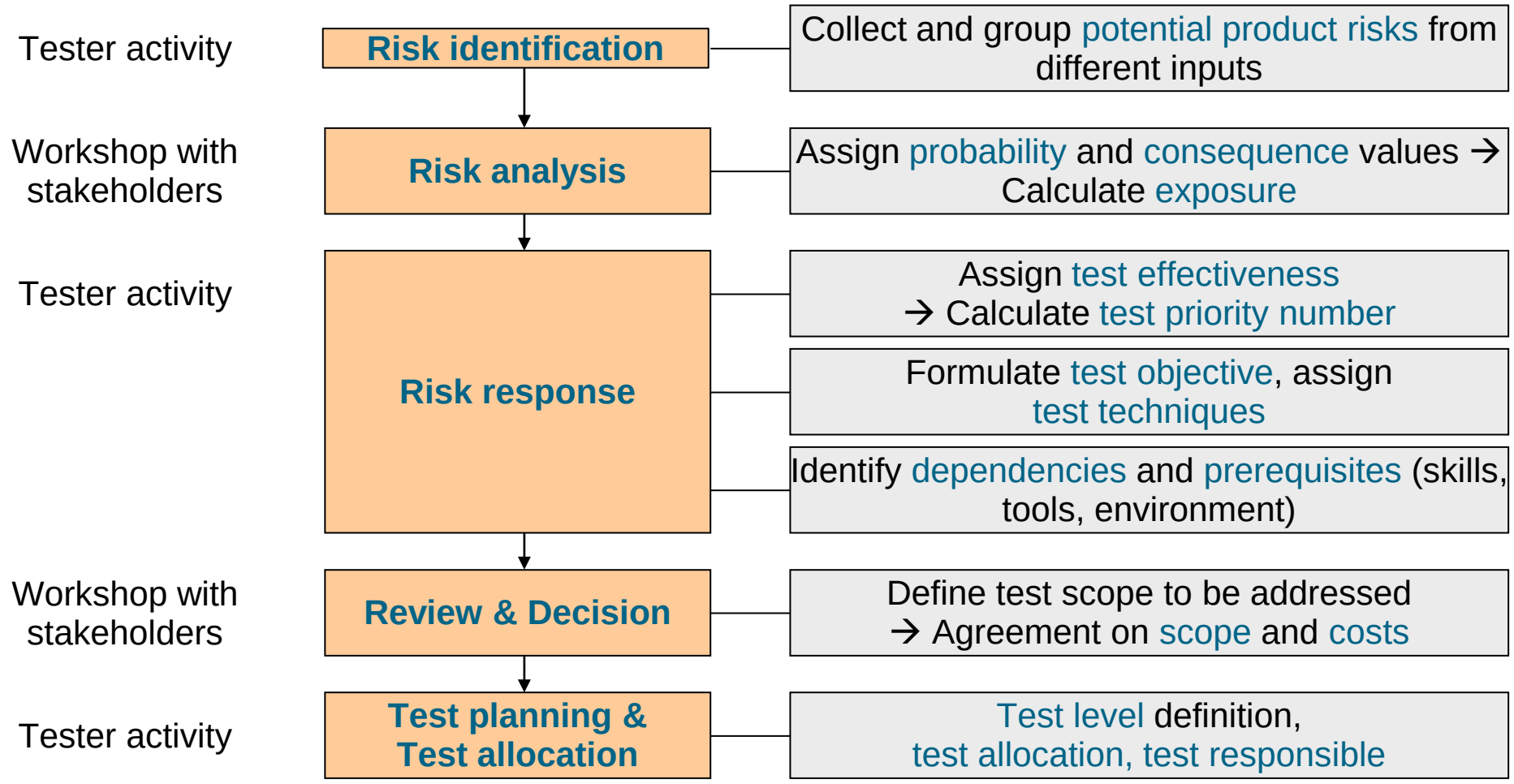
RBT is based on well-established methods

- Generic risk management techniques
  - Risk identification: brainstorming, interviews, decomposition, test basis, checklists, taxonomies, templates, questionnaires, retrospectives
  - Risk analysis and evaluation: workshops
- Failure Mode and Effects Analysis (FMEA)
  - Risk response with focus on testing
- Test planning and testing techniques
  - Test scope
  - Test planning and allocation

Example FDA

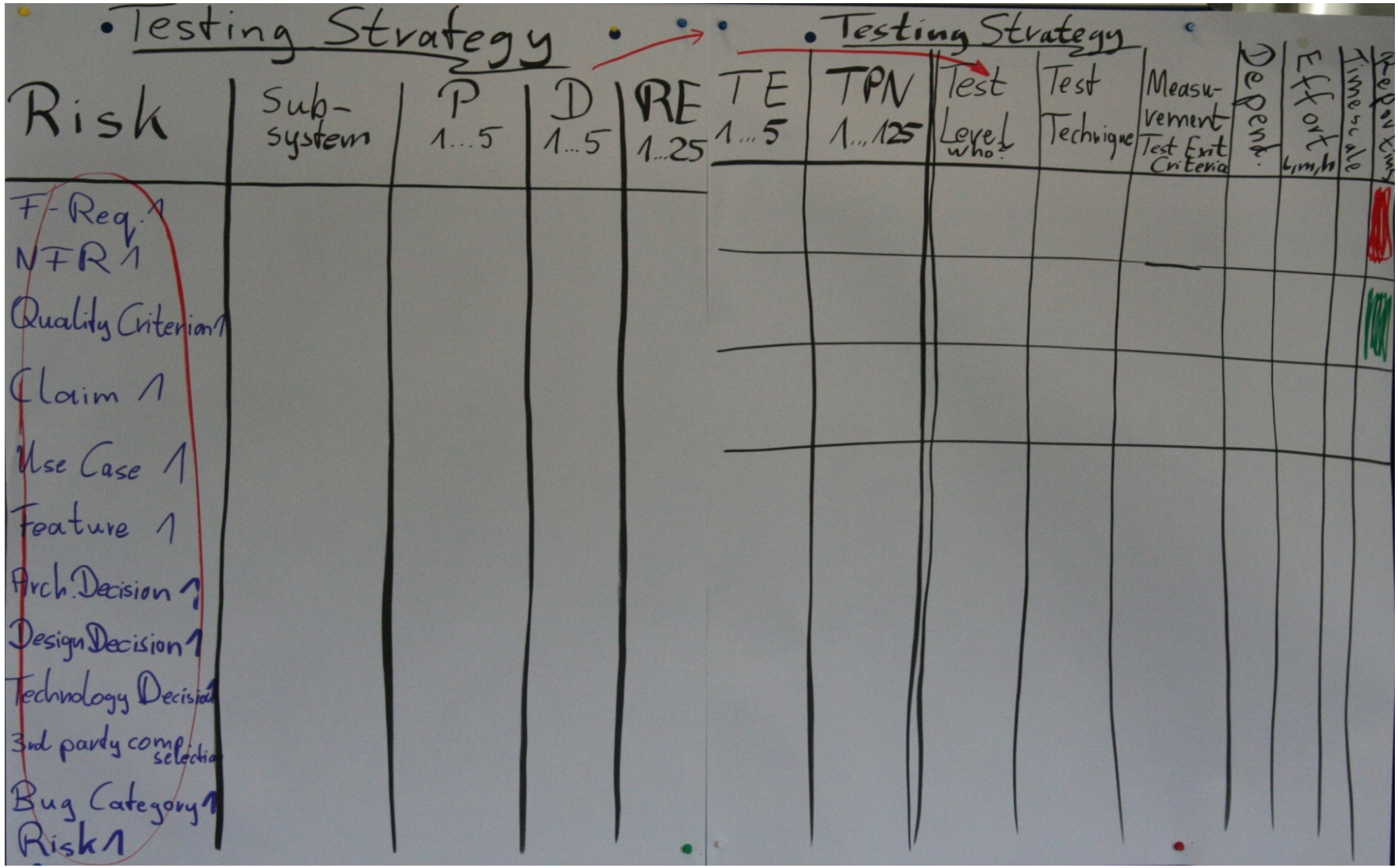


Risk-based test strategy – Steps



Adapted from: Paul Gerrard, Neil Thompson: Risk-Based E-Business Testing, 2002

Risk-based test strategy – On one page …



SIEMENS
Ingenuity for life

Risk-based test strategy (1)

Use a tabular form to develop and define the test strategy

Set up a two-dimensional matrix (rows × columns)

Rows – Items and artifacts to be considered as (product) risks

- Functional requirements and non-functional requirements
- Quality criteria (for example, based on ISO/IEC 9126, 25000), claims
- Use cases, features, functions, epics, user stories, processes, services, APIs
- Architectural decisions, design decisions
- Technology selections, third-party component selections: frameworks, tools, open source, external partnering
- Core assets in domain engineering of product line engineering
- Open variant space in software ecosystems
- Bug categories
- Risks

Risk-based test strategy (2)

SIEMENS
Ingenuity for life

Columns – Risk analysis and test planning

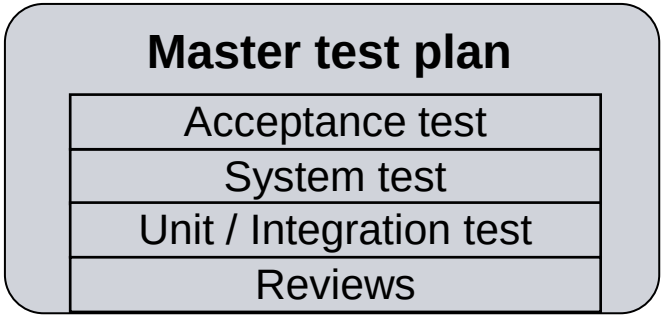
- **Risk (failure mode)**
  Description of the (product) risk – Mode of failure
- **Objective / benefits threatened**
  Which cardinal objective or business benefit(s) is threatened?
- **Subsystem**
  Which part of the system is concerned (how much) with this objective / benefit, that is, risk?
- **Probability (frequency of use, chance of failure)**
  What is the likelihood of the system being prone to this mode of failure (risk)?
- **Damage (consequence & cost for business & test & usage)**
  What is the consequence (cost) of this mode of failure?
- **Risk exposure**
  Probability × damage (consequence, cost)

Risk-based test strategy (3)

SIEMENS
Ingenuity for life

Columns – Risk analysis and test planning

- **Test effectiveness**
  How confident are the testers that they can address this risk?
- **Test priority number**
  Probability × damage × test effectiveness,
  that is, risk exposure × test effectiveness
- **Test objective(s)**
  What test objective will be used to address this risk?
- **Test level**
  In which test level is this testing performed?
  For example, reviews, unit / integration test, system test, acceptance test?
  By whom (person or group)?



Different test levels
- **Have different goals and foci**
- **Test against different artifacts of the test basis**
- **Detect different types of bugs**
- **Deliver different kinds of information**

Efficiency
- **Find bugs as early as possible**
- **Avoid unnecessary double testing**

Effectiveness
- **Do not miss important bugs by omitting a particular test level**

Risk-based test strategy (4)

SIEMENS
Ingenuity for life

Columns – Risk analysis and test planning

- **Test technique**
  What method or technique is to be used in testing?
- **Measurement**
  How can the attainment of the threatened objective / benefit – the risk reduction or elimination – be measured? For example, measurement of a quality criterion or a test exit criterion
- **Dependencies**
  What do the testers assume or depend on?
- **Effort**
  How much effort is required to do the testing?
- **Timescale**
  How much elapsed time is required to do the testing?
- **Reporting**
  Objective / benefit (not) attained – that is, risk (not) reduced / eliminated

Risk-based test strategy – Example (1)

	Identifier	Risk (Failure Mode)	Objective/Benefits Threatened	Subsystem	Probability	Damage (Consequence & Cost)	Exposure
Description	ID of entry	Brief description of the (product) risk, that is mode of failure	Which cardinal objective or business benefit(s) is threatened?	Which part of the system is concerned (how much) with this objective/benefit, that is risk?	What is the likelihood of the system being prone to this mode of failure (that is risk)? - Frequency of use - Chance of failure: criticality & complexity at implementation, criticality & complexity at usage, lack of quality	What is the damage (consequence & cost) of this mode of failure? - Consequence & cost for business - Consequence & cost for test - Consequence & cost for usage	Risk exposure, that is product of **Probability** and **Consequence (Cost)**
Scores, Ranges, and Examples			For example: Quality criteria	Scores from 1 to 5: 1 Very low importance ... 5 Highest importance	Scores from 1 to 5: 1 1-20% Highly unlikely, chances are slight 2 21-40% Unlikely, probably not 3 41-60% We doubt, improbable, better than ever 4 61-80% Probable, likely, we believe 5 81-99% Almost certainly, highly likely	Scores from 1 to 5: 1 Negligible: no noticeable effect 2 Low: business will be affected slighlty 3 Moderate: business objectives will be affected 4 High: business objectives will be undermined 5 Critical: business objectives cannot be accomplished	Range between 1 and 25
1	Functional requirement						
2	Non-functional requirement (NFR)						
3	Quality criterion						
4	Claim						
5	Use case						
6	Feature						
7	Function						
8	Epic						
9	User story						
10	Process						
11	Service						
12	API						
13	Architectural decision						
14	Design decision						
15	Technology selection						
16	3rd party component selection (frameworks, open source, external partnering)						
17	Core asset in PLE						
18	Open variant space in software ecosystems						
19	Bug category						
20	Risk						
21							

Risk-based test strategy – Example (2)

Test Effectiveness	Test Priority Number	Test Objective(s)	Test Level	Test Technique	Measurement	Dependencies	Effort	Timescale	Reporting
How confident are the testers that they can address this risk?	Product of **Probability**, **Consequence**, and **Test Effectiveness** that is product of **Exposure** and **Test Effectiveness**	What test objective will be used to address this risk?	In which test level is this testing performed? By whom (person or group)?	What method or technique is to be used in testing?	How can the attainment of the threatened objective/benefit, that is the risk reduction or elimination be measured?	What do the testers assume or depend on?	How much effort is required to do this testing?	How much elapsed time is required to do this testing?	Objective/Benefit attained, that is risk reduced/ eliminated
Score from 1 to 5: 1 Testing is not the way to address this risk or an appropriate test objective would prove to be unachievable ... 5 High confidence that testing will find faults and provide evidence that the risk has been addressed	Range between 1 and 125	For example: demonstrate that... verify that... validate that... check that...	For example: unit testing integration testing system testing acceptance testing developers integration test group system test group	For example: black-box testing white-box testing	For example: a measurement for a quality criterion a test exit criterion	For example: a test entry criterion	For example: high medium low	For example: days weeks months	Objective/Benefit not attained, that is risk not reduced/ eliminated

SIEMENS
Ingenuity for life

Risk-based test reporting – Concepts

Traceability

- Reorganize metrics of testing,
  i.e. for example bug status and test pass / fail status,
  based on their relationship to risks

Risk weighting / prioritization

- Risk Exposure = Probability × Damage

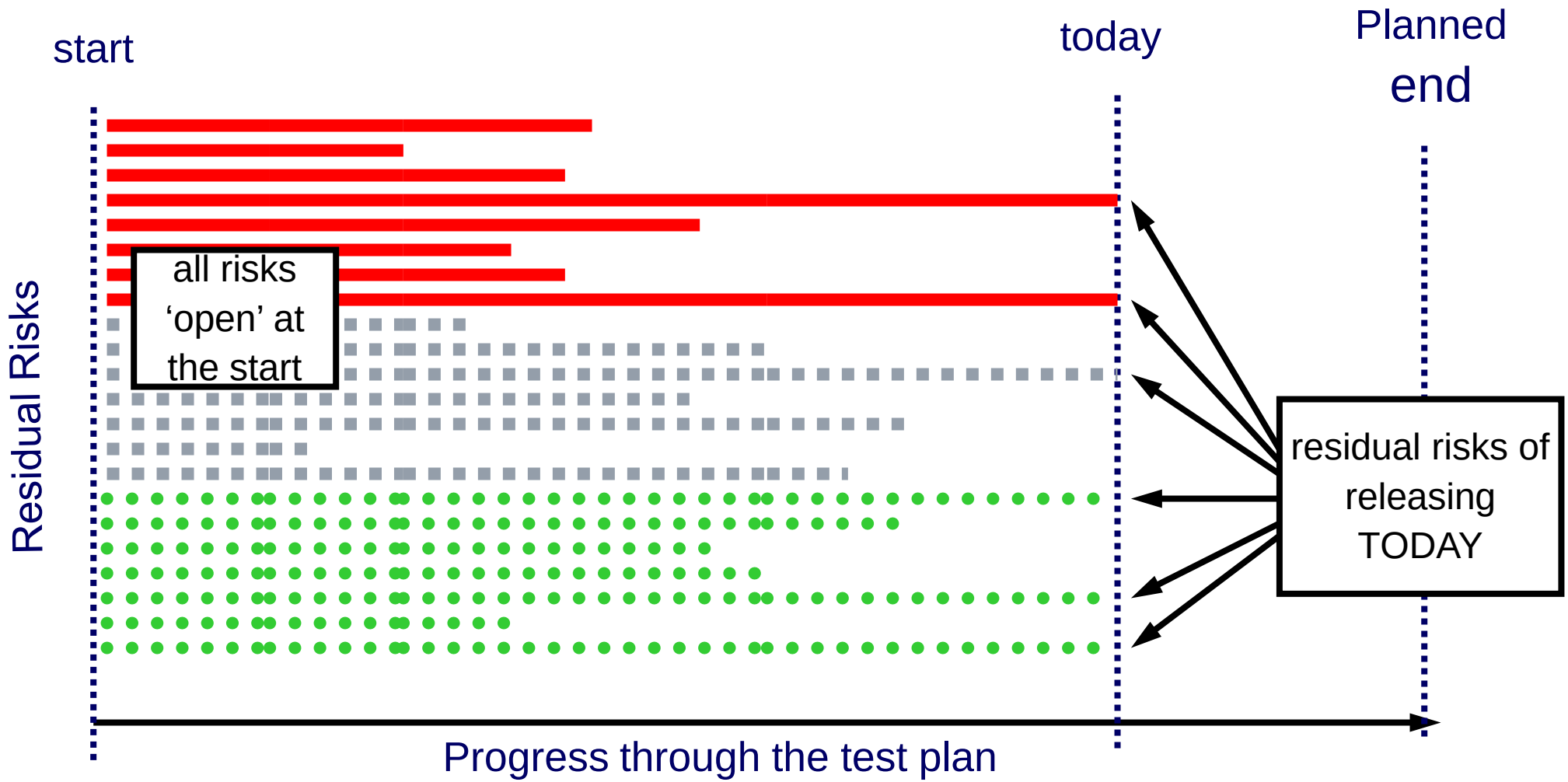
Risk status classification

- Use status on bugs, test progress, and coverage to classify the status of each risk item in different groups

Triage of *new features versus technical debt*

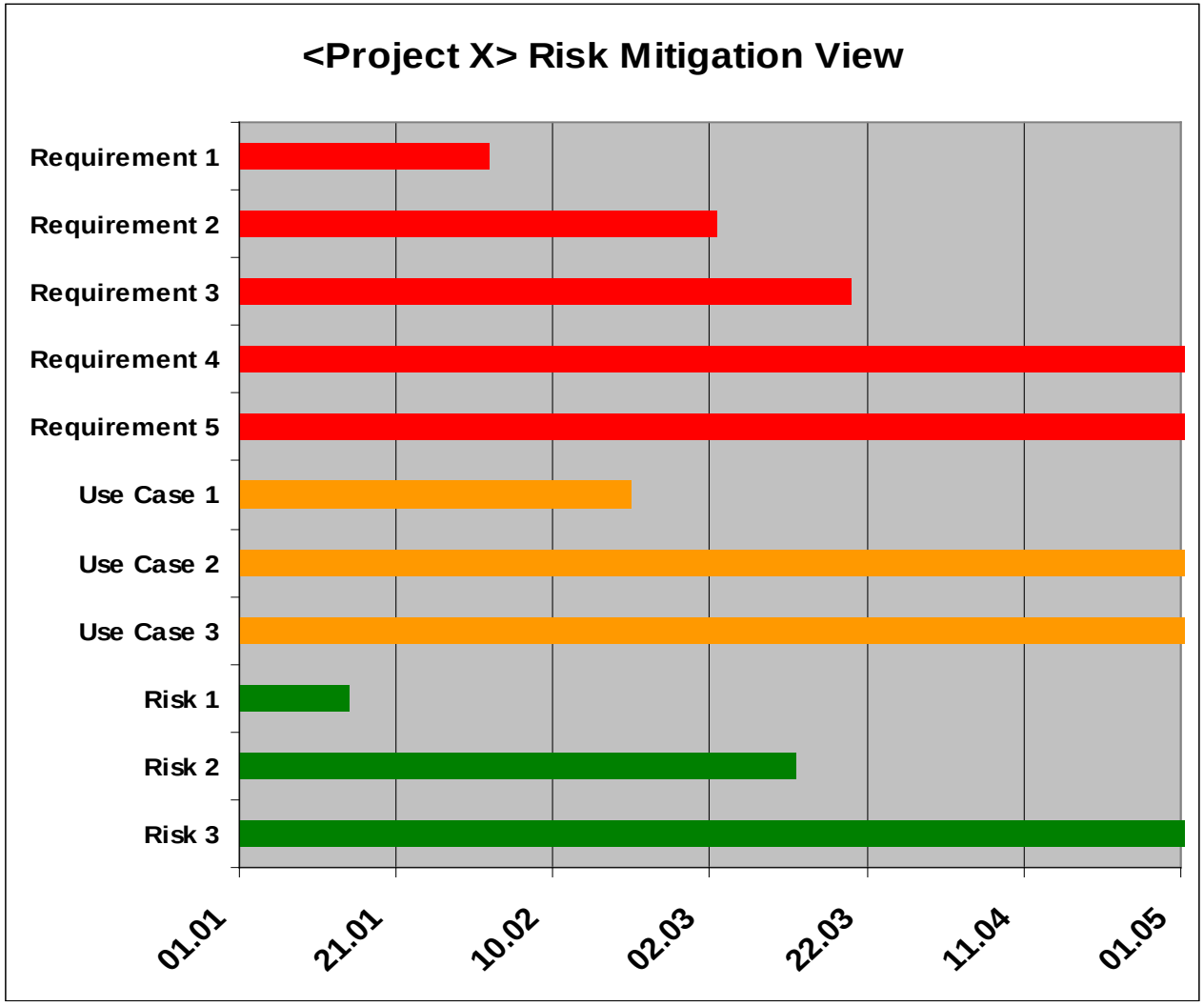
Ship early with confidence – residual risk indicates readiness for ship

Risk-based test reporting – Example

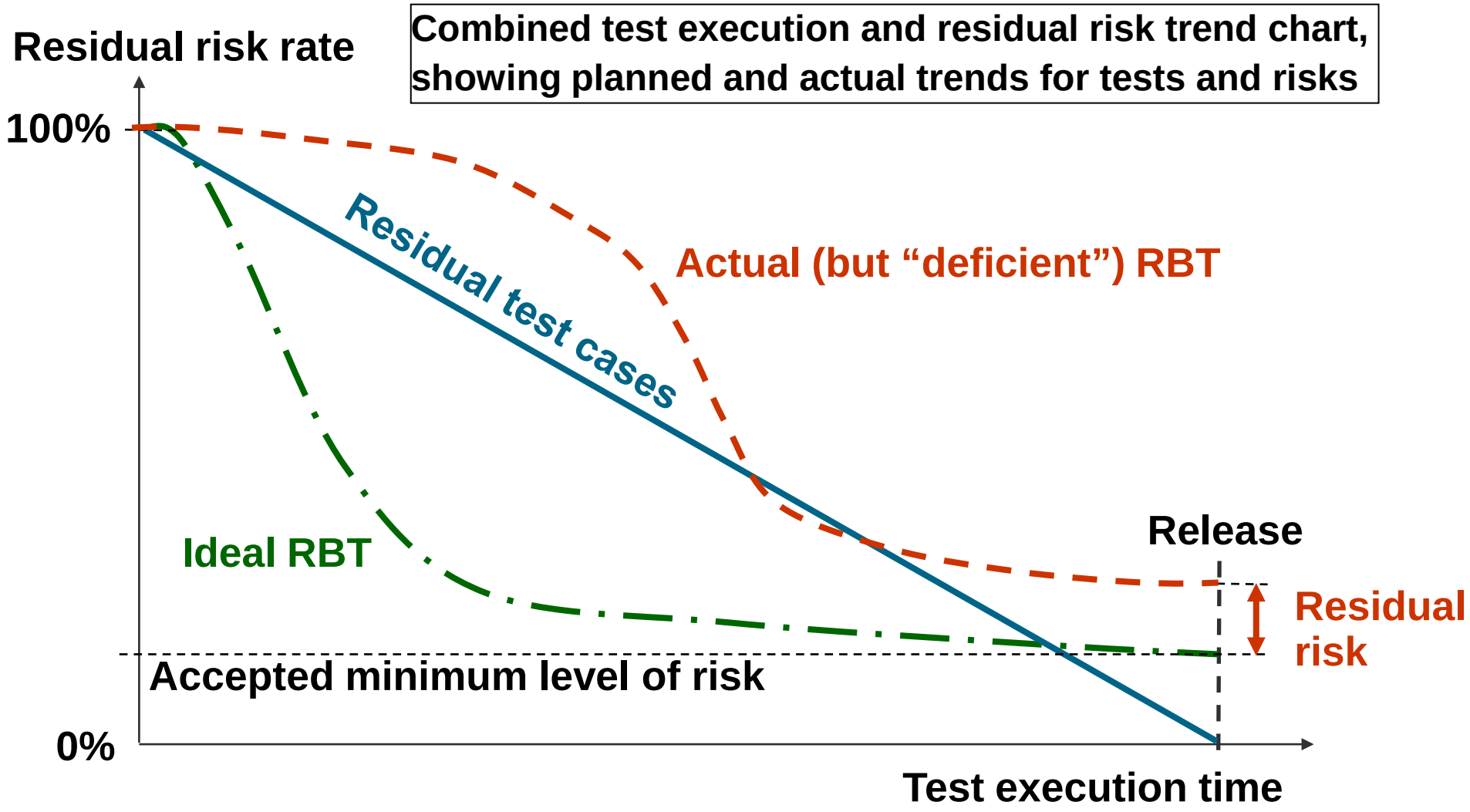


Reference: Paul Gerrard

Risk-based test reporting – Example



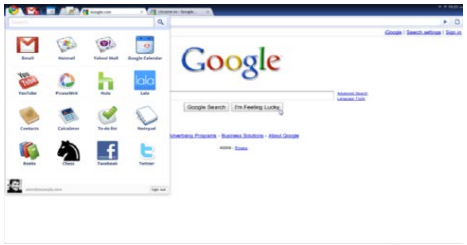
Residual quality risk trend chart



Example – RBT @ Google – Chrome (Browser + Frame + OS)

SIEMENS
Ingenuity for life

Identified matrix of product attributes, components, and capabilities
About 300 tracked items
Attacking top risks first
Risk drives what is tested (not dev or shiny objects)



Top risk areas for Chrome OS

- Networking
- AutoUpdate
- Browser/Desktop UX
- Out-of-Box-Experience
- Login/Account scenarios
- Media Player
- Power management/states
- Flash/Plugins
- Profile sync
- I18N

Start Page | Mail | Calendar | Documents | Web | more ▾ — Report a bug | sebs@google.com ▾ | Settings ▾ | Sign out

Google docs Chrome OS Risk Analysis — People at Google.com with the link — Saved 2 minutes ago — Saved — Share

File Edit View Insert Format Form Tools Help — sebs is viewing

	A	B	C	D	E	F
1	Attribute	Component	Capability	Estimated Frequency of Failure	Estimated Impact to User	Capability Risk
2	From Sheet 1	From Sheet 2	Behavior of the component in response to the attribute	A choice of (very rarely, seldom, occasionally, often) {1,2,3,4}	A choice of (minimal, some, considerable, maximum) {1,2,3,4}	Calculated automatically
121	Simple, Elegant	Talk Video	gmail / igoogle interaction with panels	3	3	9
122	Simple, Elegant	Talk Video	Chat moles in page	3	3	9
123	Simple, Elegant	Talk Video	Simultaneous video encoding + decoding	3	3	9
124	Simple, Elegant	Talk Video	Make, receive phone calls of good quality	2	2	4
125	Simple, Elegant	Talk Video	Launch app from contacts	2	2	4
126	Internationalization	Sync	Language and input method, IME dictionaries sync with cloud	3	2	
127	Secure	Sync	Secure sync of system settings and data (including network/password info)	2	2	4
128	Simple, Elegant	Sync	Sync service handles on/off availability of the sync server	3	4	12
129	Simple, Elegant	Sync	Sync settings provide control of which data to sync	2	3	6
130	Stable	Sync	Recovers from corrupted sync of settings/data (including network/password info)	2	4	8
131	Secure	Server Side	Matrix of hardware and OS image versions and update checks (Omaha)	3	4	12
132	Secure	Server Side	Internal Glass support & access via web browser	3	3	9
133	Secure	Server Side	Remote wipe of machine	2	4	8
134	Secure	Server Side	OS image & updates deployment and availability (Lorry)	2	3	6
135	Gaia	Security/TPM	Login - Seamless Gaia integration	4	4	16
136	Gaia	Security/TPM	Login - Support full set of Gaia accounts (gmail, googlemail, dasher)	3	3	9
137	Gaia	Security/TPM	Login - Promiscuous & incognito mode	2	3	6
138	Gaia	Security/TPM	Login - User account should be locked after Nth failed password attempt	2	3	6
139	Internationalization	Security/TPM	Login - L10n & i18n during login & account management	3	4	12

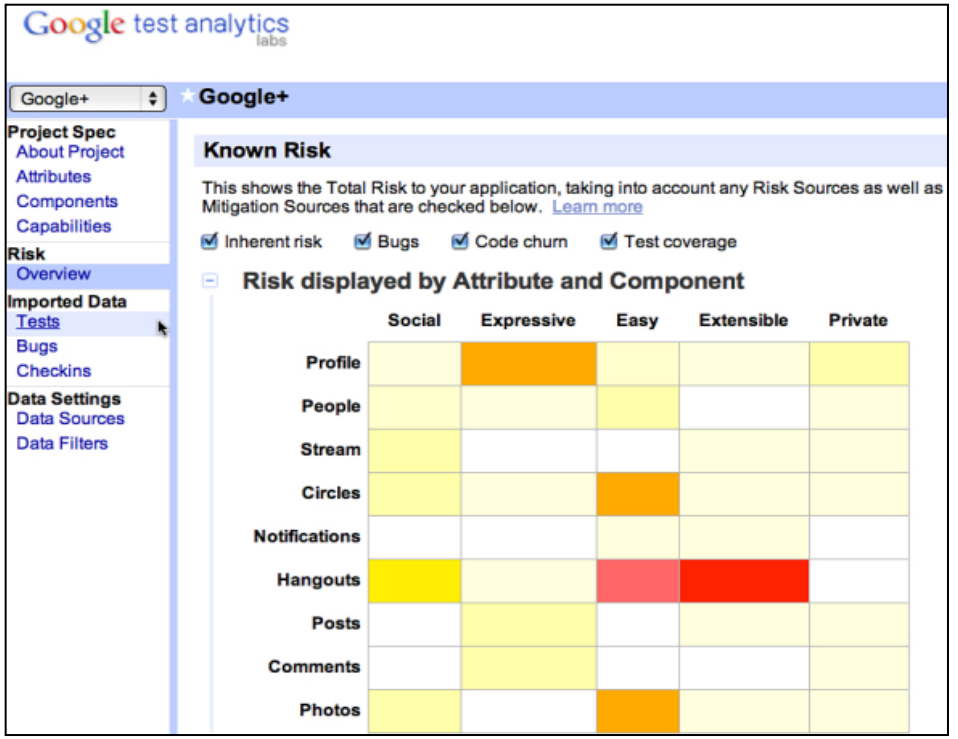
Reference: Jason Arbon, Jeff Carollo, Sebastian Schiavone (Google), StarEAST Conference 2010

Example – RBT @ Google

ACC methodology for test planning and RBT

- **Attributes –** Marketing speak (business justification): adjectives, adverbs
  Qualities of the product that differentiate it in the marketplace → why?
- **Components** – Parts of the system (code): nouns
  Artifacts from how the system is designed → what?
- **Capabilities** – Features: verbs
  What the system can do, intersection of **a**ttributes and **c**omponents

Risk is the *frequency* of failure of a **Capability** multiplied by the *impact* it will have on the user

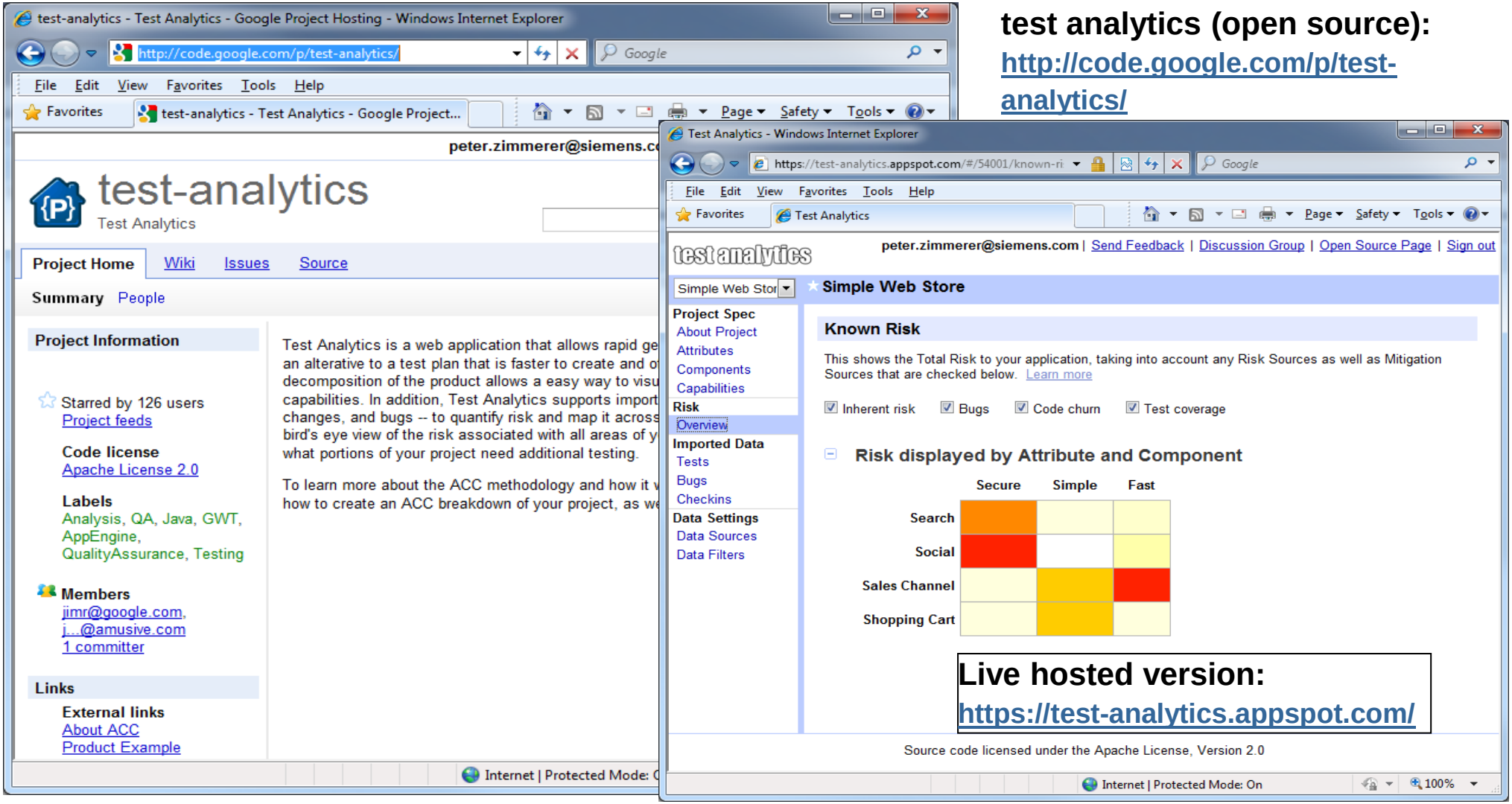


Reference: James A. Whittaker, Jason Arbon, Jeff Carollo, How Google Tests Software, 2012

Example – RBT @ Google – Google Test Analytics (GTA)

SIEMENS
Ingenuity for life

test analytics (open source):
http://code.google.com/p/test-analytics/



Lessons learned regarding RBT

Start risk-analysis by doing a proper stakeholder analysis

State the product risks in a business language

Recognize that impact and likelihood are different

Visualize the results of the product risk analysis

Consider both functional and non-functional risks

Define a differentiated risk-based test approach

Report against the identified product risks

Choose the product risk analysis method that meets your needs

Revisit the product risks analysis on a regular basis

Establish clear risk ownership and responsibilities

Reference: Erik van Veenendaal, StarTester Issue 53, August 2010

Test Strategies

Agenda

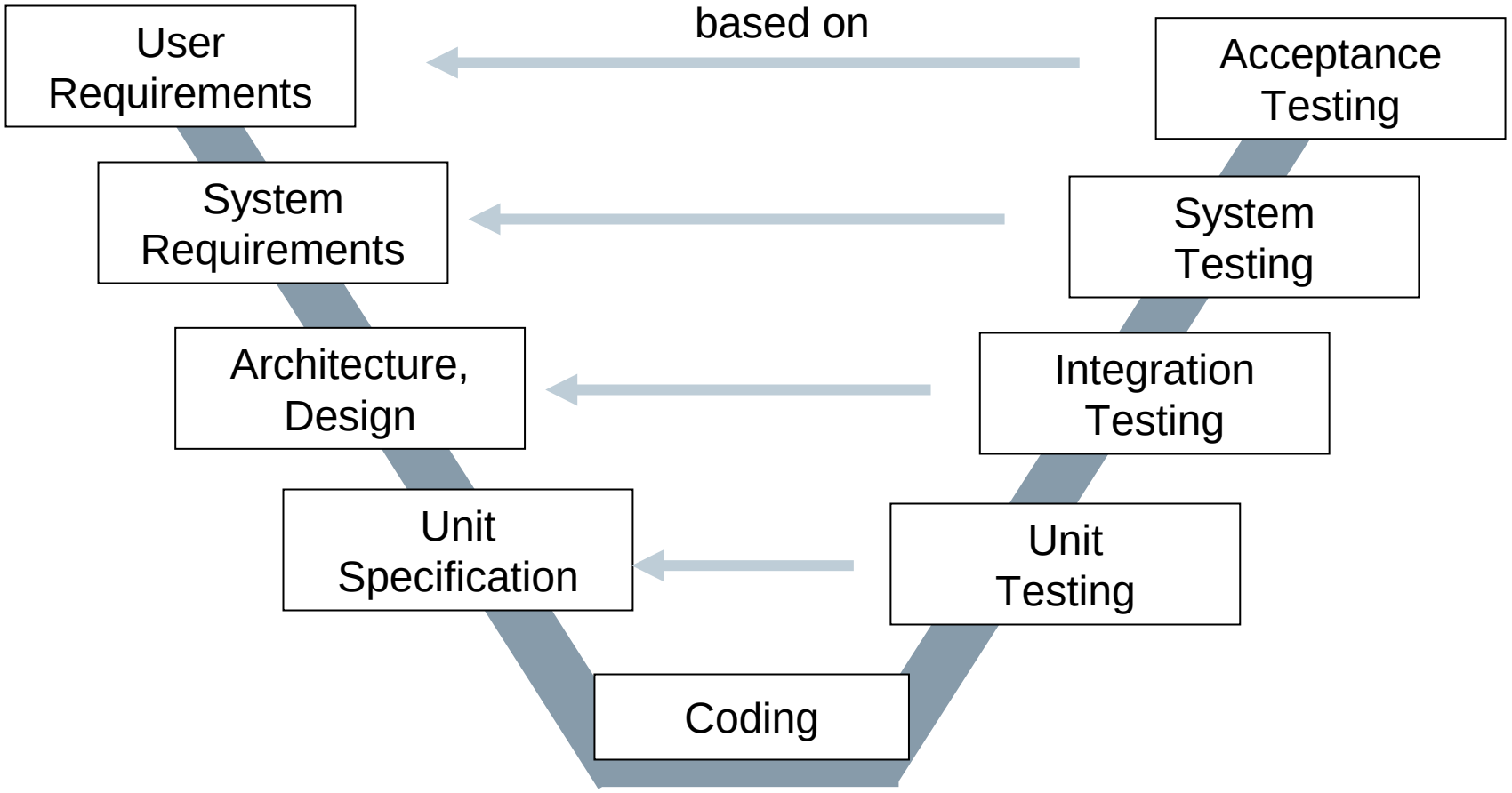
Strategies for Testing

Risk-Based Testing

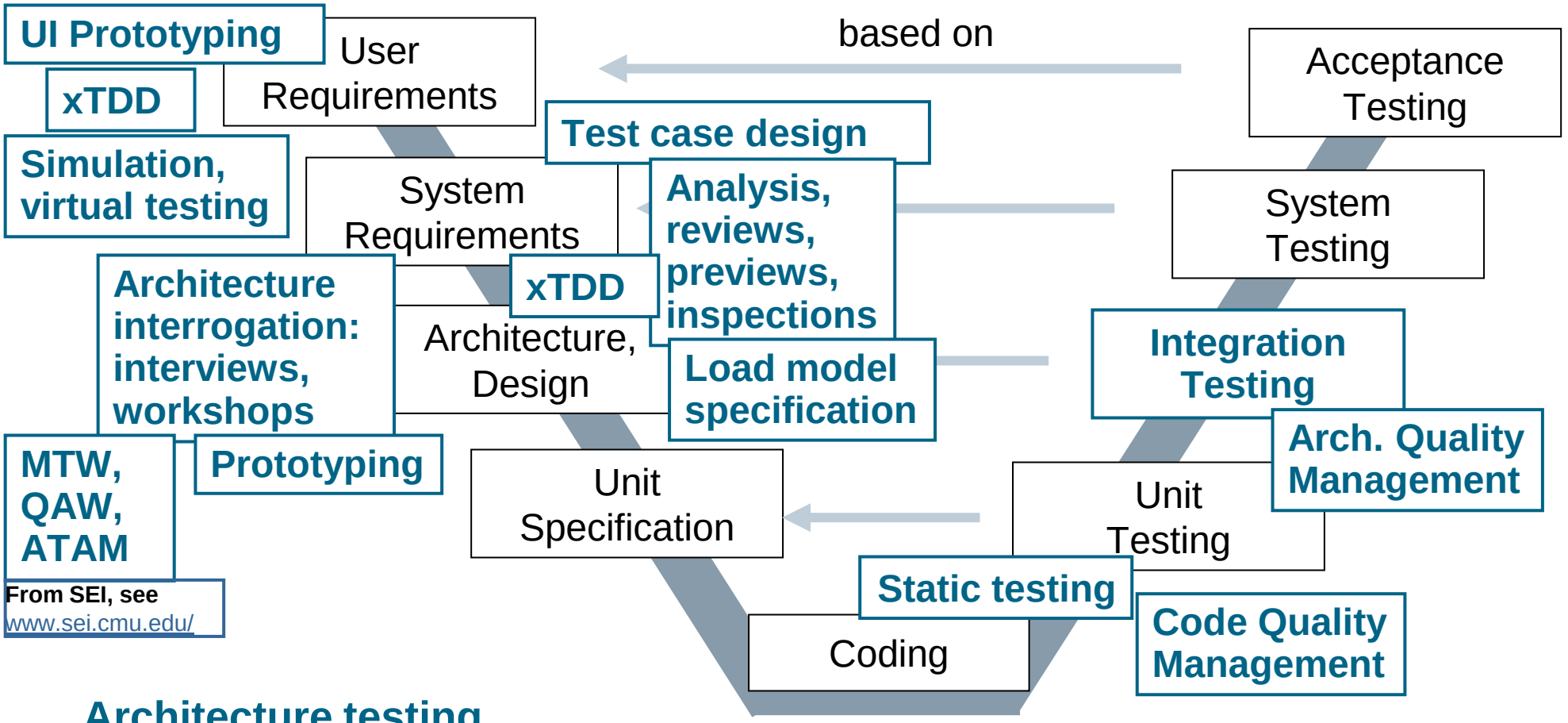
Test Levels & Types

Summary

Test levels – Example V model



Test levels – Example V model with architecture testing



Architecture testing

is any testing of architecture and architectural artifacts

→ mapping of architectural risks to test levels

Architecture testing@ Google Search

SIEMENS
Ingenuity for life

Result hits* @ Google Search:

Requirements testing	**287,000**
Architecture testing	**60,800**
Document testing	**41,500**
Unit testing	**4,900,000**
Integration testing	**1,680,000**
System testing	**2,530,000**
Acceptance testing	**2,290,000**

*Hits counted on May 29, 2013 – ratio still about the same since 2005 …

Integration testing (1)

IEEE Standard 610.12-1990
Testing in which software components, hardware components, or both are combined and tested to evaluate the interaction between them.

BCS SIGiST Testing Standards Working Party (http://www.testingstandards.co.uk/living_glossary.htm)
Testing performed to expose faults in the interfaces and in the interaction between integrated components.

TestWorks (http://www.soft.com/Technology/glossary.html)
Exposes faults during the process of integration of software components or software units and it is specifically aimed at exposing faults in their interaction

Integration testing (2)

ISTQB Glossary (http://www.istqb.org/)
Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems. See also *component integration testing, system integration testing.*

Component integration testing
Testing performed to expose defects in the interfaces and interaction between integrated components.

System integration testing
Testing the integration of systems and packages; testing interfaces to external organizations (e.g. Electronic Data Interchange, Internet).

SIEMENS
Ingenuity for life

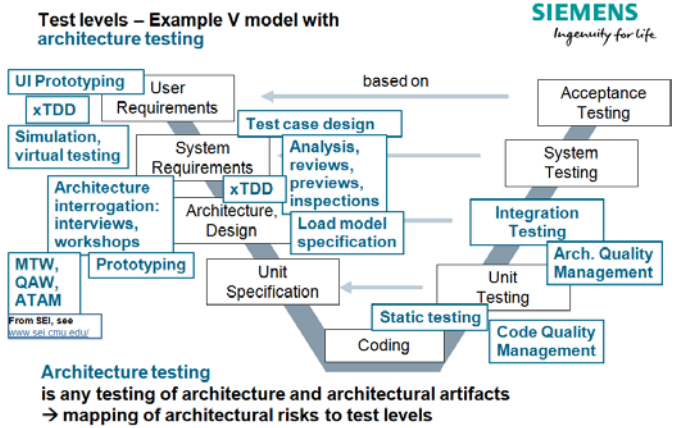
Integration bugs

- Use of incompatible parts (configuration / version control problem)
- Protocol design errors
- Missing, overlapping, or conflicting functionality
- Incorrect or inconsistent data structure used for global data
- Conflicting data usage
- Violations of data integrity of global data
- Wrong method called due to coding error or unexpected runtime binding
- Client sending a message that violates the server's precondition (violated design by contract)
- Client sending a message that violates the server's sequential constraints
- Wrong object bound to method (polymorphic target)
- Wrong parameters, or incorrect parameter values
- Failures due to incorrect memory management allocation / deallocation
- Failures due to missing resources, missing synchronization
- Failures due to interaction with 3rd party components / software (→ ecosystems)

Integration testing

An integrated system is a complex equation, built from interdependent variables. Explore these variables and their interactions.

Len DiMaggio, 2003



The goal of integration testing is to test in a grey-box manner

- Interaction of components, parts, processes, subsystems
- Interoperability and embedding with the environment and system configurations
- Interfaces and dependencies (explicit, implicit)
- Dynamic behavior and (internal) communication of the system (states and events)
- Control flow and data flow, data consistency and integrity
- Architecture / Design / API conformance
- Architecture and design as specified in the **S**oftware **A**rchitecture **D**escription document (starting point: architecture vision)

During integration test execution the internal behavior of the system under test is monitored by using tracing facilities to provide the required information

Integration – Integration testing

Integration

Constructive: Integration of components / sub-systems to get a running system (or parts)

Quick-check by performing a **smoke test** only for **most important** interfaces and functions

Periodical delivery of integrated system (or parts) corresponding to the integration plan

Successful integration is a precondition for **handover** to next step / level (integration testing)

Integration testing

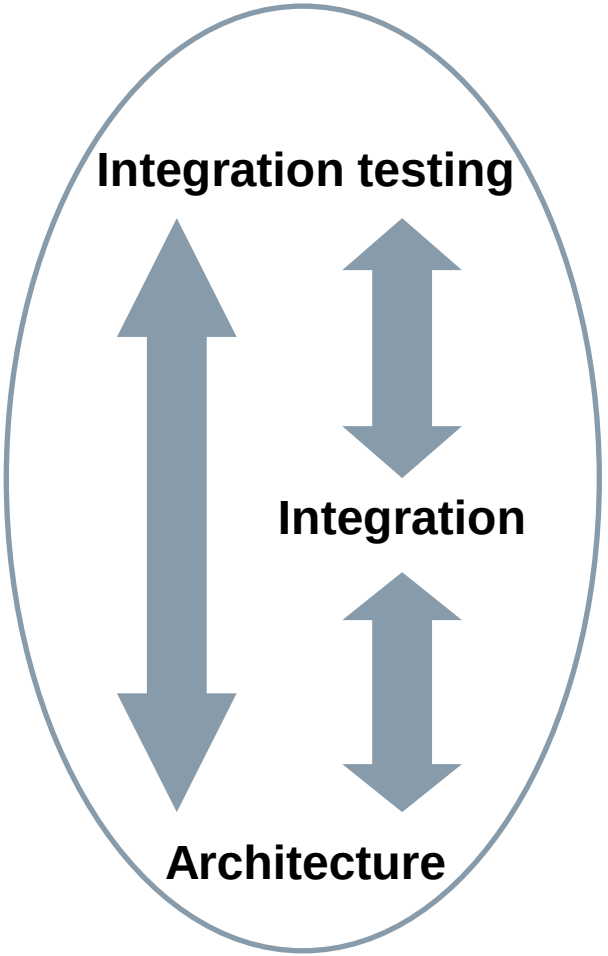
Quality assuring: Test of the system to detect bugs in the interworking of its parts, to gain confidence, to mitigate risks, etc.
Detailed check of the system by performing a **systematic** test of **all** interfaces and functions corresponding to the test concept
Progressive test as a separate testing level corresponding to test concept with defined begin and end dates

Successful integration testing is a precondition for **handover** to next step / level (system testing)

Integration test strategies

Integration testing follows the integration strategy which is based / depending on the architecture

Big bang ←→ Incremental
Top down ←→ Bottom up
Architecture-driven based on functional threads Function-wise, service-based Layer-wise, graph-based Backbone Skeleton
Based on risk, criticality, importance, priority Availability, based on delivery Fully functional parts Based on easiness, heuristics



Integration testing levels

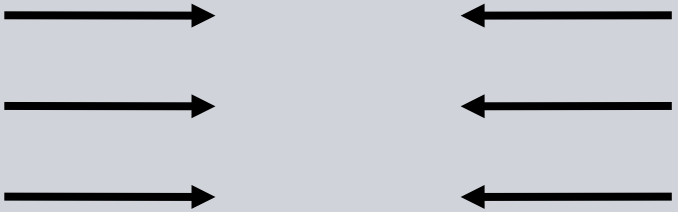
There may be more than one level of integration testing on a project

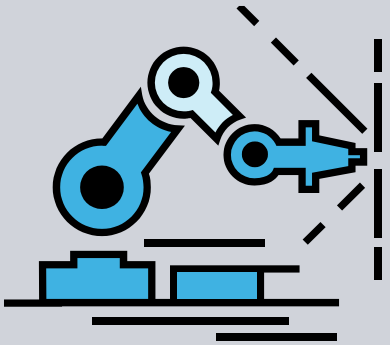
- Component integration testing: look for bugs in interactions between units or components following unit testing
- System integration testing: look for bugs in interaction between entire systems following system testing

System integration testing is complex

- Multiple organizations controlling the systems' interfaces, making change dangerous
- Business processes may span systems
- Hardware / system compatibility issues can arise

Example: Integration Testing





Control system
- Set operation mode
- Control robot actions
- Visualize status

Requests
- Init/Shutdown
- Move to position
- Stop moving
- Get operation status

Replies
- ACK/NACK
- Return state

Robot system
- Operational states
- Error states
- Real time constraints
- Safety constraints

- What aspects of this system would you test on **Integration Test** level?
- What kind of input would you need for these tests?
- Which stakeholder(s) can provide these inputs?
- What do you – as **Test Engineer / Test Architect** – have to provide?

Integration testing – System testing (1)

Integration testing	System testing
Test against the **architecture / design** of the system: tries to find design errors	Test against the **(user) requirements** of the system
Test goal: Validation of **parts** of the system	Test goal: Validation of the **whole** system
Test focus: Functionality **(grey-box)** at internal and external interfaces of the architecture / design, performance, robustness	Test focus: Functionality **(black box)** at the GUI, usability, configuration-, load-, stress-test
More idealistic test environment possible (e.g. simulation environment)	**Realistic** test environment needed (hardware, devices, etc.)
Needs more **technical** know-how	Needs more **(end-) user** know-how

Integration testing – System testing (2)

So, in contrast to integration testing, system testing

- Uses an already completely integrated system
- Performs full black box tests from a user's point of view, for example based on use cases
- Does not take into account control or data flow information or design specifications
- Concentrates on the behavior of the whole system, but does not specifically test a single interface, component, etc.
- Does not consider the detailed internal behavior of the system under test (for example message sequence calls, parameter values, etc.)

Test types

A/B test	API-Test	Availability test	Built-in test
Compatibility test	Concurrency test	Configuration test
Conformance test	De-installation test	Documentation test
Endurance test	Ergonomics test	Function test
Fuzz test	G11N (Globalization)	GUI test
Installation test	Interface test	I18N (Internationalization)
Interoperability test	Load test	L10N (Localization)
Memory mgmt test	Passive testing (tracing / monitoring)	
Performance test	Random test	Real-time test
Recovery test	Regression test	Reliability test
Reuse test	Robustness test	Scalability test
Security test	Simulation	Smoke test
Stability test	Stress test	Usability test

Test Strategies

Agenda

Strategies for Testing

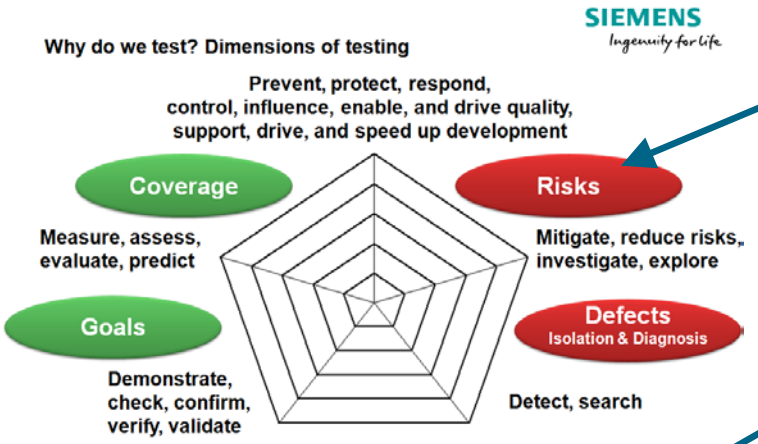
Risk-Based Testing

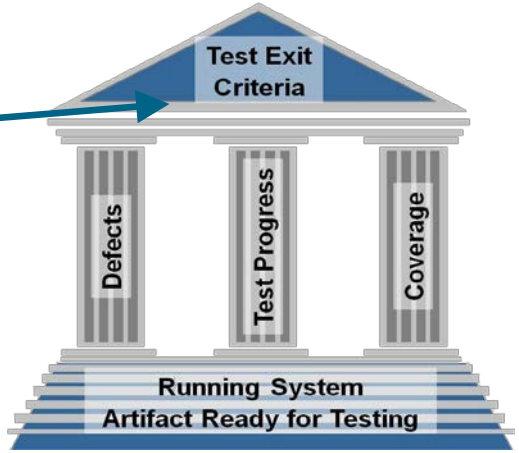
Test Levels & Types

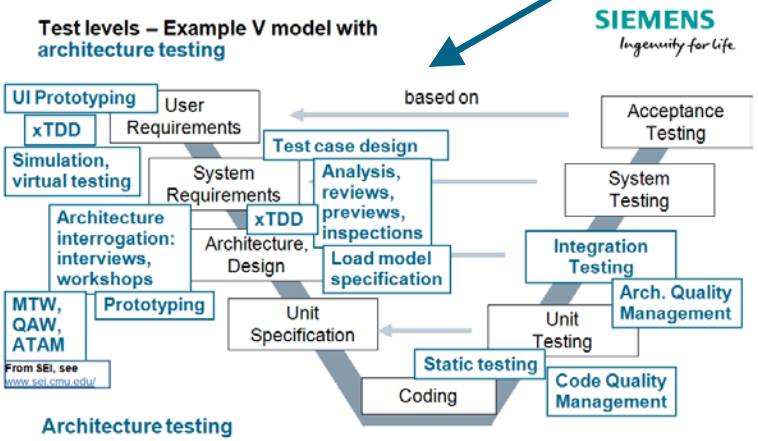
Summary

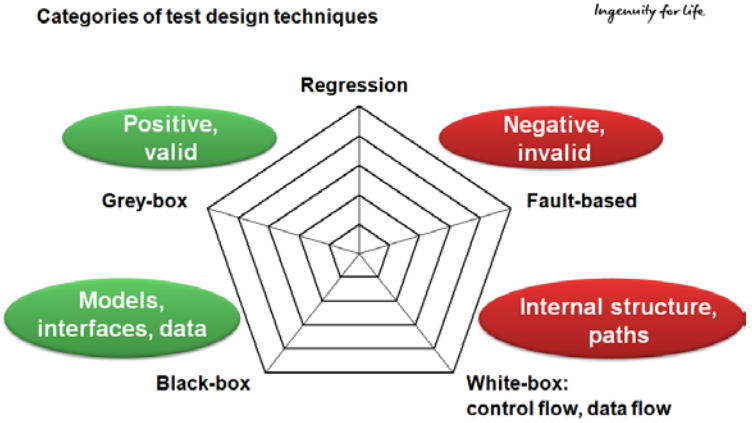
Summary: Building blocks of a *Risk-based Test Strategy (RBT)*

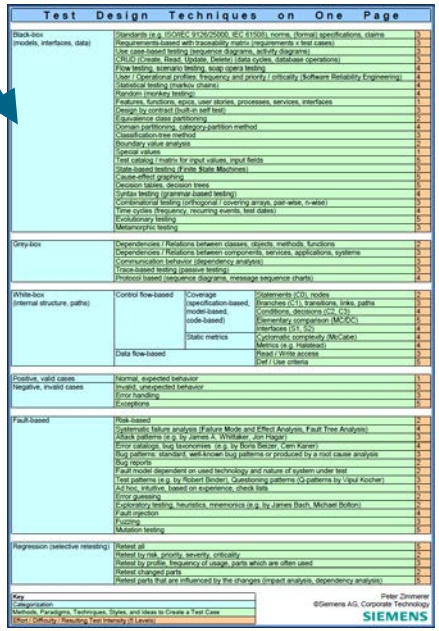
SIEMENS
Ingenuity for life











What we have learned

The test strategy defines what, why, when and how to test throughout the lifecycle

Risk is the main driver for the test strategy
→ No risk, no test!

Risk identification and analysis is the responsibility of all stakeholders; testers can facilitate and drive this process.

Several, different test levels ensure effective and efficient testing over the lifecycle.

Integration testing follows the integration strategy which is based on the architecture.

Departing thought

If you want to build a high-quality test architecture,
don't drum up people to
collect requirements and design specifications,
divide the testing work,
and give orders.
Instead, teach them to yearn for
a sustainable (test) architecture with quality inside
and satisfied clients.

[Freely adapted from Antoine de Saint-Exupéry]

Further readings

Use the SSA Wiki :
https://wiki.ct.siemens.de/x/fReTBQ

and check the “Reading recommendations”:
https://wiki.ct.siemens.de/x/-pRgBg

- Architect's Resources:
  - Competence related content
  - Technology related content
  - Design Essays
  - Collection of How-To articles
  - Tools and Templates
  - Reading recommendations
  - Job Profiles for architects
  - External Trainings
  - ... more resources