Quality is the responsibility of the whole team

- Test do not create quality or take it away, it is only a method of assessing the quality of a product.
- If development schedules get tight, make sure project managers does not use the test team as quality gatekeeper at the end of the development process by skipping qa activities and testing at earlier stages of development.
- Include test coordinators in status meetings or make sure they get all information that may influence the scope, prioritizations, method and/or progress of testing activities including test planning.

Never trust the test team to be the quality gatekeeper

How to make the testers love you (tester=business expert)

- Configuration management procedures are not invented to make developer's life miserable – lack of configuration managements makes everybody miserable
- System test is the end users first encounter with the new system – think about how you can contribute to a GOOD first impression
- Insist on project resources for deliverables and ways of working that promotes good quality (business term glossary is one example, coding standards and reviews are others)
- Defects that should never survive unit test and unit integration test are
  - Invalid input allowed (e.g. alphanumeric input in numeric field, values out of range)
  - Missing error messages
  - Missing field labels, buttons, values in drop-down lists etc.

Test tools – Computer Aided Software Testing (CAST)

Topics

- Type of test tools
- Test automation
- Selection and introduction of test tools

Why invest in test tools ?

Some reasons why an enterprise needs software testing tools are:

- The demand for testing is continuously growing
  - accelerating marked demands for new releases of business systems
  - increasing number of user interface platforms and
  - the need to test upgrades of a whole range of technical infrastructure components.
- Can make testing cheaper

There just is not enough time, money and human resources with the right knowledge to keep testing coverage at an acceptable level using only manual testing.

- The business systems of today are so complex and diversified that it is impossible to assess the quality unless one has a tool to support the test process and keep track of all quality attributes and their statuses.
- Multi-site organisations with stakeholders, tester and developers spread over several sites and even in different time zones also counters for a tool to ease sharing of information.
- Can make testing more systematic

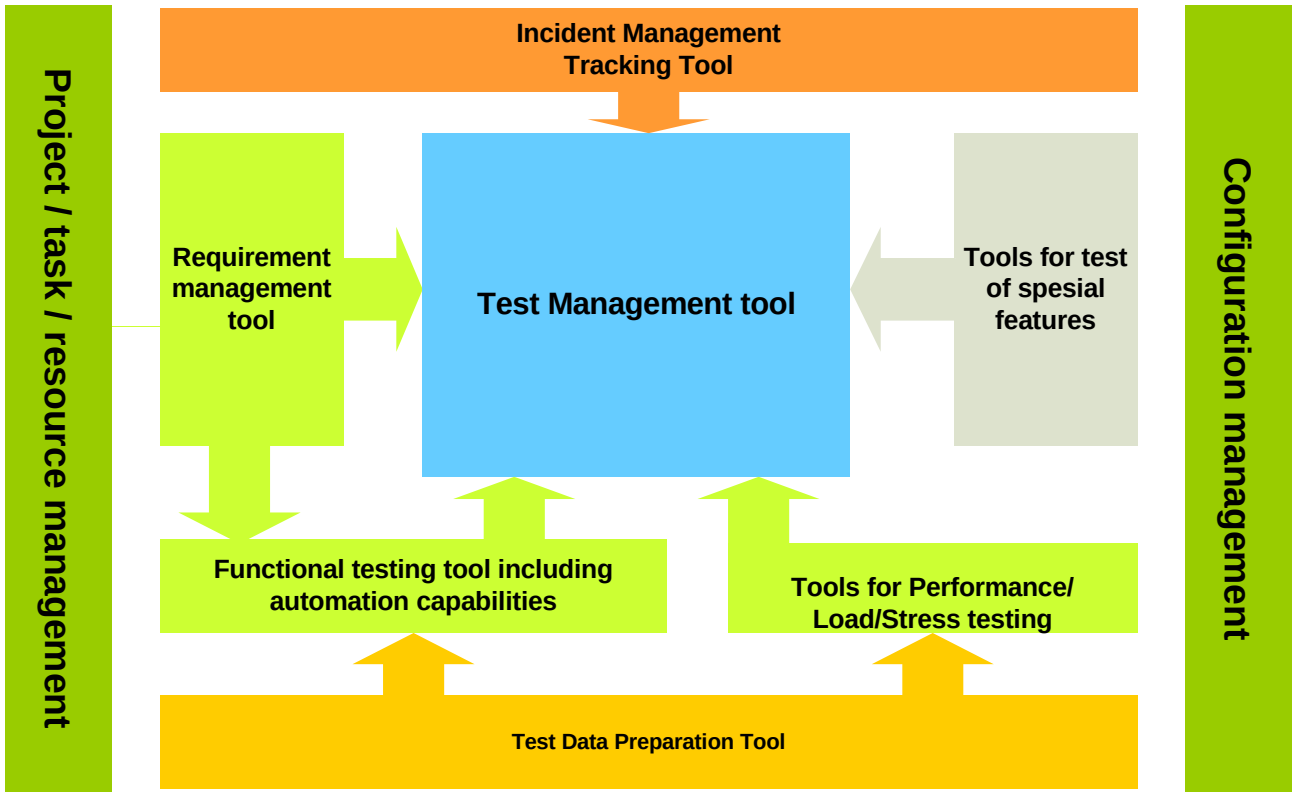
- Some tests are not possible to do without tools

ALM = Application Lifecycle Management

- The term is commonly used for products that somehow touch the application lifecycle, from requirements gathering and configuration management to project management and monitoring.
- New tools and improved versions of existing tools are continuously introduced to the market.
- No vendor support ALL aspects of ALM in one single tool

A myriad of vendors and tools

ALM tool model



One key requirement to a test tool is OPEN INTERFACE capabilities.

Configuration management

- **The most important tool !**
- Lack of configuration management is the 8th deadly sin!
- A developer
  - must always know which version he is deploying
  - must always keep in mind changes in different branches of the code
- A tester must always know
  - which version of a product he is running his tests against
  - use the corresponding version of test script/procedure

- There is seldom a good excuse for bad configuration management. Still a lot of developers and testers everywhere waste a lot of time due to lack of version control.

Tool for test of special features

- Security testing (data privacy, and access control, virus checking)
- Usability testing
- Coverage tool
- Website link checking tool to look for broken links
- Drivers for early verification of interface specifications
- Dynamic analysis tool (debugger is one example, performance testing tool another)
- Static code analyzer
- Cross-browser compatibility tool

Some basic terms

- Static testing tool: traverse and check the code without executing it
- Dynamic testing tool: executes the code
- Test framework:
  - A tool to make execution of a test object possible
  - A developer tool to isolate a component from its surroundings (or that replaces or generates drivers and stubs)
  - Otherwise used in industry for:
    - Reusable and extensible testing libraries to build testing tools (such tools are also called ”test harness”)
    - A type of design of test automation (like data-driven, keyword-driven)
    - The process for executing tests
- Test automation framework
  - defining the format in which to express input variables and expected results
  - creating a mechanism to hook into or drive the application under test
  - executing the tests
  - reporting results

Test management tool

- Support the whole test process: test planning, test design, test execution and incident management.
- Test planning (coverage, status of test scripts (design, review, ready), who is responsible, priority
- Test design
- Test execution planning: test cycles, test sets, when and whom
- Interface for executing tests and logging test results.
- Progress and status reports (which tests are passed, failed, not completed, not run)
- Team collaboration (mail notification is one example)
- A lot of these tools also support requirement management:
  - Requirement prioritization, release planning, version handling
  - Risk models to support risk based testing

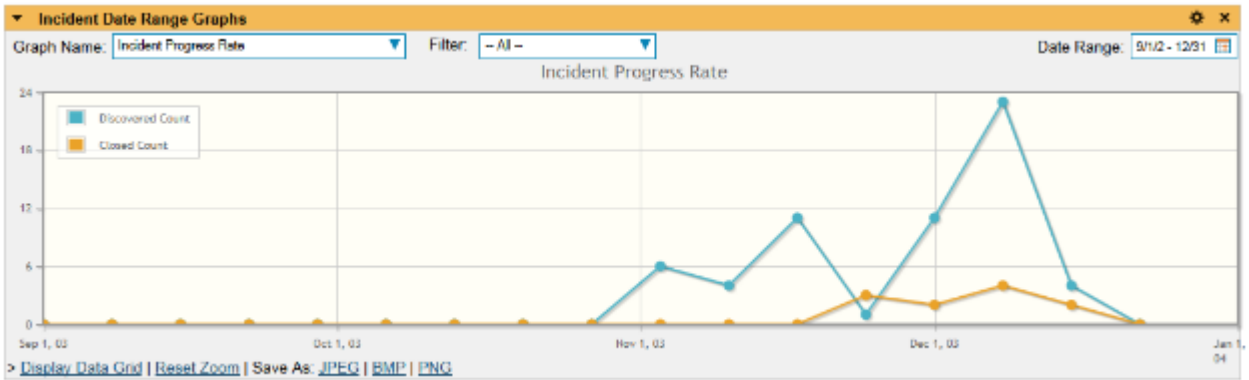
Test management tool requirements

- Provide necessary metrics for quality assessment including requirement traceability
- Support multi-projects
- Support agile as well as waterfall software development lifecycle models
- Support distributed teams (‘My page’ is a very useful feature!)
- Integration with tools from other vendors, e.g. tools for test automation
- Ease of use
- Ease of customization (roles, work flows, access rights, attributes)
- Customer support
- Innovation
- TOC – unless your budget is unlimited

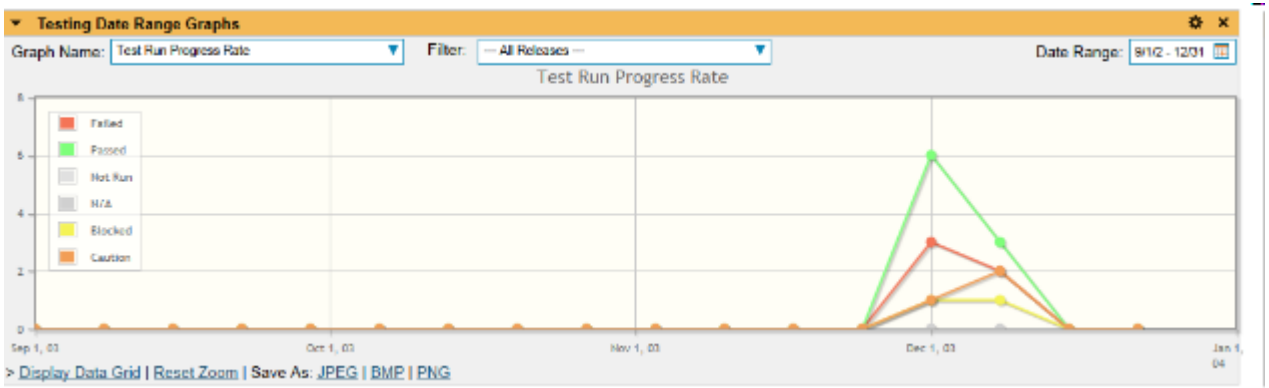
Using test results to assess quality

- Metrics from test:
  - Requirements coverage summary: requirements verified OK, requirements failed and requirements not yet tested.
  - Requirement incident count
  - Summary of test execution status;
    - Number of test passed, number of tests failed, number of tests not run.
  - Incident summary report showing total number of incidents grouped by severity and status.
  - A list of all unresolved incidents
  - Incident progress rate together with test run progress rate

Graphs supporting test completion decision



Incident Progress Rate Graph



Test Run Progress Rate Graph

Potential benefits of test management tool

- Ease of access to information on test procedure tasks and test results
- (Less) Unbiased assessment of quality
- Reuse of test scripts is simplified
- Test process improvement

NOTE!! *Using a test management tool is no guarantee for high quality deliveries*

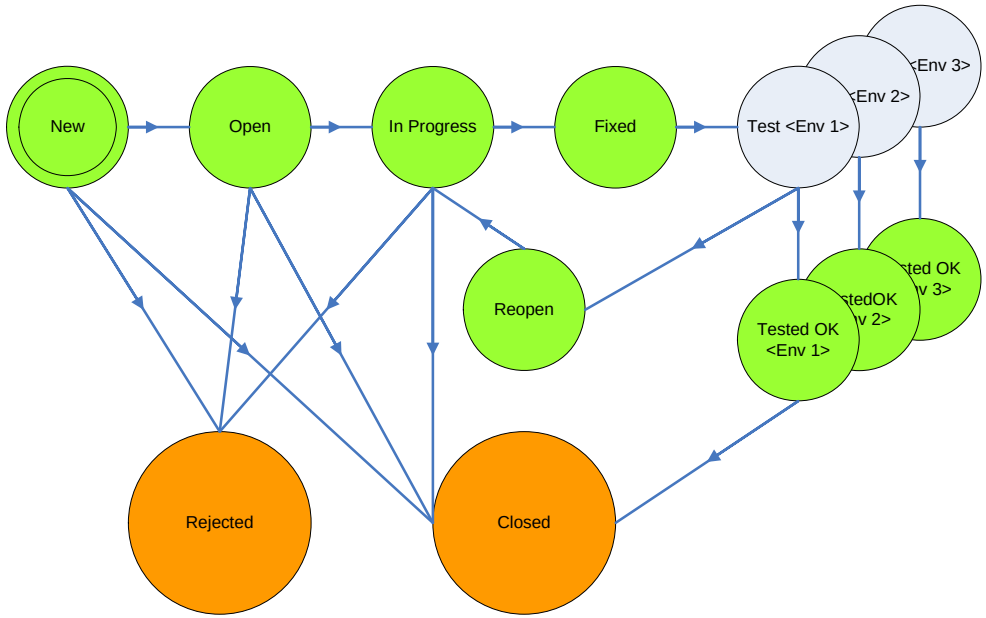
- Metrics from the tool can show complete requirements coverage, 100% successful test execution and no open defects at test termination and still result in a product release with a lot of defects. This is normally caused by a combination of several factors like
  - Incomplete requirements
  - Inadequate test data
  - Poor test design
  - Test not focusing on the high risk areas
  - Testing tasks left out of plans because not enough time and resources
  - Inadequate version handling of requirements, code and test assets
  - -…..
- These factors are due to procedural shortcomings, lack of software testing skills etc. and can not be helped by any tool

Incident management tool

- Incident reports with documentation such as screen captures
- Categorization
- Prioritization
- Allocation
- Support several incident types (defect, change request, issue…)
- Incident work flow depending on incident type
- Role based access rights (not all project members should be allowed to close a defect)
- Statistical analysis (can for instance spot error prone application areas)

- Integration with configuration management (which files are modified when fixing a defect)

Incident workflow example



Valid Transitions and roles

New -> Open: Bug fix manager when assigning issue to developer
Open -> In Progress: Developer when starting work on the incident
In Progress -> Fixed: Developer when defect is ready for deployment, assigns to deployment
Fixed -> Test <env 1>: Deployment assigns to Test Manager when fix is deployed in <Env 1>
Test <Env 1> - >Tested OK <Env 1>: Tester assigns to Deployment or to Test Manager if Env 1 is 'last stop'
New, Open, In Progress, Reopen -> Rejected : Test Manager, Bug fix manager or developer assigns to tester

Tools for test design and test data generators (From Hans)

- From database
  - Generates databases and files from schema
  - Filters contents from existing bases
  - Translates data between different formats
- From code
  - In order to cover all code
  - Problem: Missing code not found
  - Problem: Code itself is test basis :-(
- From interfaces
  - Finds which data are used in interfaces
  - Can make a GUI-map
  - Can generate erroneous input
- From (requirements) specification or models
  - Formal notation necessary
  - UML, MSCs, state charts
  - “Model based testing”
- Generators for combinations
- Random generation

Creativity of a test designer cannot be replaced by tools. But “standard tests” can be generated.

Expected results/ test oracle: Very difficult to generate.

Tools for static testing

- Tool that support the review process (plan, execute and evaluate results)
- Static analyzers (developers tool) (See chapter on static testing)
  - The compiler
    - Data flow and control flow anomalies
    - Code standard violations
    - Field boundaries over-/underflow
  - Measurement of code complexity to select code that needs review
- Model checking tools

- Website link checking tool to look for broken links
- These kind of tools are cheap and using them can save a lot of time and money!

Test automation

- Test automation has for more than a decade been praised as the silver bullet to control the increasing cost of software testing.
- The myth that automation can reduce head count creates a dilemma when initiators for automation requires resources for test automation planning, script generation and maintenance of scripts.

- Return on investment for test automation cannot be measured in head counts but rather in these potential benefits
    - the ability to find defects up front
    - better test coverage with the same number of testers
    - less boring work
    - better accuracy and repeatability
    - the ability to uncover defects that manual testing cannot (like system resource overutilization)

Things to consider when planning test automation

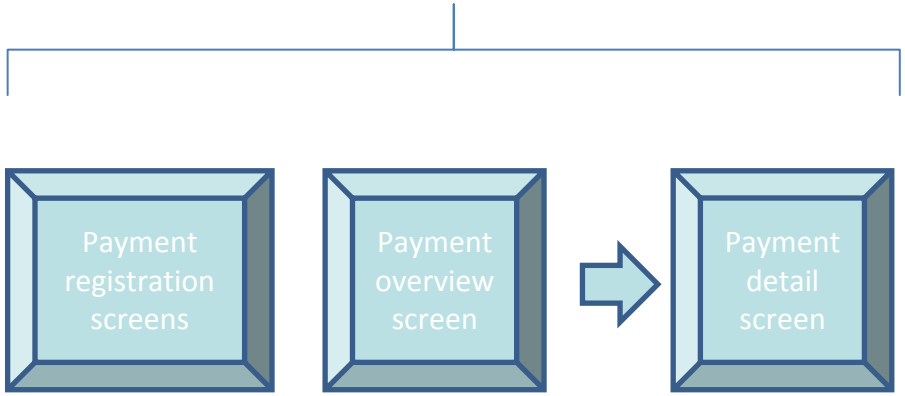
- How is your manual testing process?
- **The principle of finding defects as early as possible is valid also with automated testing**
- Submitting untested software to automated system test is very bad for the ROI of automation

> **"If you automate chaos, you get faster chaos."**
>
> (Dorothy Graham)

Planning- Factors to consider when picking the best candidates for test automation

- Stability of test object
  - Detailed knowledge of test object and preferably also the road map for later releases
  - Frequency of manual testing
- Complexity
- Test Data
  - Test consumes data and it is no practical way of a refresh
  - It must be possible to restore test precondition automatically either by roll-back or by running a script ahead of the test
- Risk

Test of payment – a test automation example



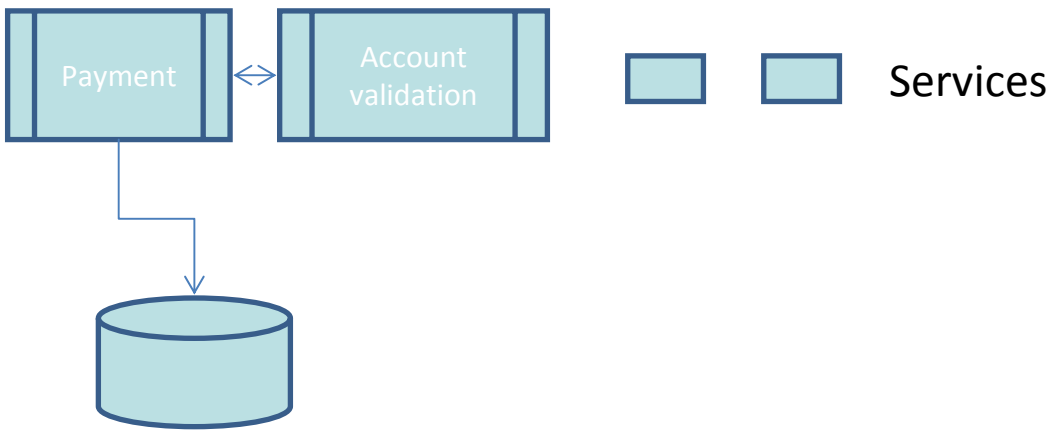
Input:
Debit account
Credit account
Payment date
Payment amount
…..

Lists Future payments

Display payment details

Test script flow:

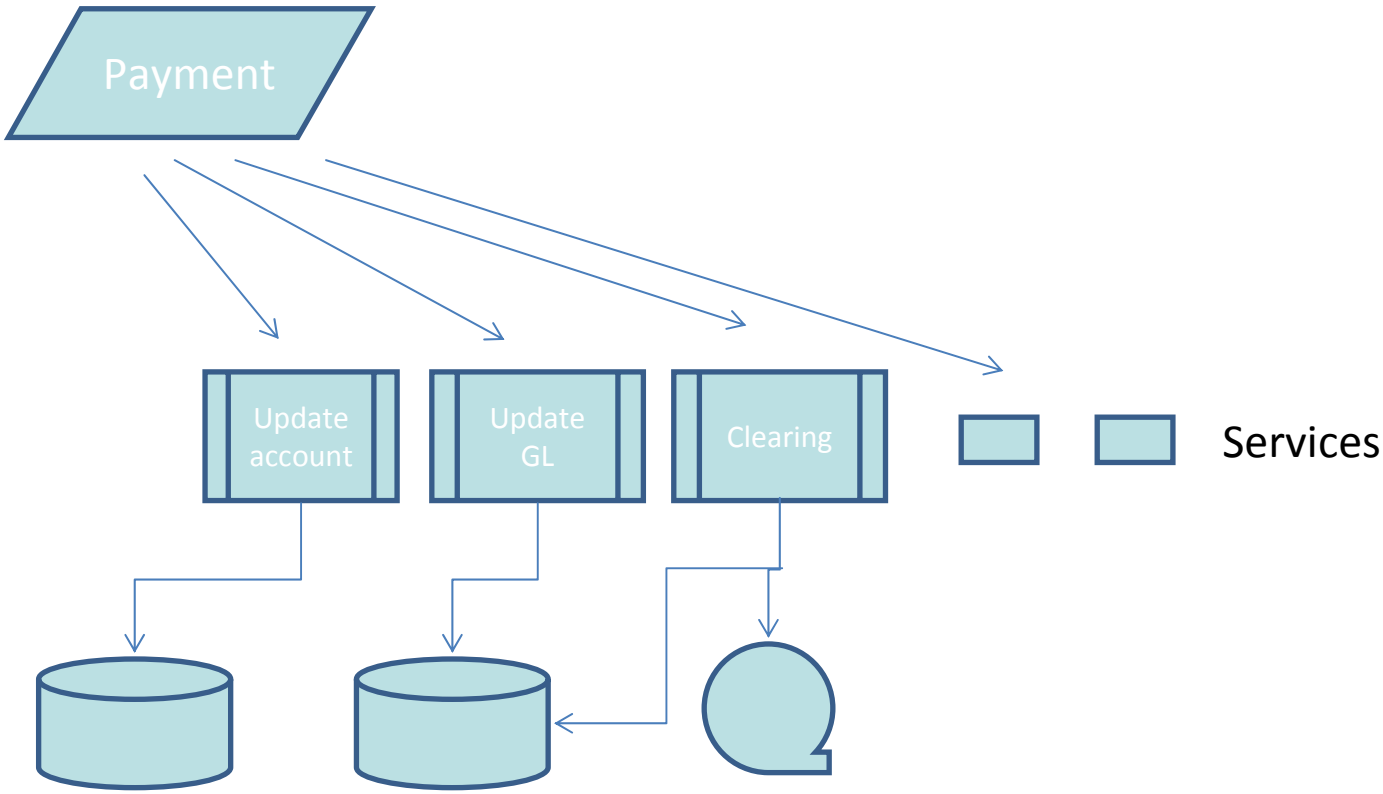
1. Register payment
2. Go to payment overview
3. Select payment from step 1
4. Compare all fields with input from step 1
5. Return to payment overview
6. Cancel payment from step 1



Test script design details:
Payment screen is tested in every detail:
Invalid input
(date, amount, account, missing input, message field and so on….), use of tabulator key, use of buttons …

Problems with test design in previous slide

- The test pretends to be a system test/acceptance test. Business testers can be mislead to think that all business rules related to payments have been validated.
- Defects related to update of account balance, update of GL will remain undetected.
- In order to validate remaining business process, testers will have to enter a new payment
- Repeating tests on screen input will eventually not detect any new defects



Soft aspects related to test automation

- Business testers mistrust automatic test results – want to be assured by checking themselves
- Business testers feel automation is a job threat
- Business testers are intimidated by the technical harness surrounding automated scripts and feel they loose control of the test process

Neglect of these aspects can stop the automation initiative.

Test automation initiative and organization

- Involve business testers early in test automation initiative.
- Involve business testers in test planning and design.
- Communication: You cannot expect the business expert to read 10 pages of technical test design documentation. Think carefully about how new testers shall get sufficient information about the test coverage.
- Be accurate about which testing still has to be manual
- Status reports must be a summary of manual and automated test coverage and execution results so business experts testers get the full picture both during planning and during execution.
- Tool support must be adequate during build up of automated test suit but also afterwards in ‘everyday use’ of the test suite.
- Run a health check of test environment up front before running the business test scripts. Testers will mistrust scripts that often fails due to non-functional errors and revert to manual testing

Test automation – different tool types

- Capture/replay tools
  - Are very exposed to changes in test object
  - (But can be useful for performance testing)
- Pure script based tools -Writing programs to test programs:
  - Development resources
  - Test scripts also needs debugging
  - Test scripts also need maintenance
  - The challenge of keeping the test scripts aligned with the test object
- Key word driven test design (action-word) tools
- Data driven testing tools
  - Anything that has a potential to change (also called "Variability" and includes such as environment, end points, test data and locations, etc), is separated out from the test logic (scripts) and moved into an 'external asset'. This can be a configuration or test dataset in a spreadsheet or database. The logic executed in the script is dictated by the data values.
- A lot of tools on the market combine some of these characteristics

Keyword driven tools

- **Pros**
  - Maintenance is low in a long run
    - Test cases are concise
    - Test Cases are readable for the stake holders
    - Test Cases easy to modify
    - Keyword re-use across multiple test cases
  - Not dependent on Tool / Language
  - Division of Labor
    - Test Case construction needs stronger domain expertise - lesser tool / programming skills
    - Keyword implementation requires stronger tool/programming skill - with relatively lower domain skill
  - Abstraction of Layers.
  - Scripts can be used for both manual and automatic test execution
- **Cons**
  - Longer Time to Market (as compared to manual testing or record and replay technique)
  - Moderately high learning curve initially

Cost of automation

- Training
- Script
- Design (unless you can reuse test design for manual testing)
- Documentation
- Test data
- Incident analysis (Do not underestimate this – incidents are produced at a much higher rate. When a test fails, is it caused by a setup bug, test data bug, automation bug or product bug)
- Test execution
- Change management and maintenance
- Servers and other infrastructure costs
- License fees and tool maintenance fees

Calculation of ROI for test automation

- Common formula for calculation of winnings

Manual testing cost =

Manual preparation cost + cost of manual test execution **x** times executed

Automated testing cost =

Automation preparation cost + cost of automated test execution **x** times executed

- Several flaws with this approach
  - You cannot count tests that you normally will not repeat with manual testing
  - You cannot equate manual testing with automated testing
    - The automated test does everything exactly like every time and will eventually not find more defects
    - The human tester has awareness of all kind of product characteristics

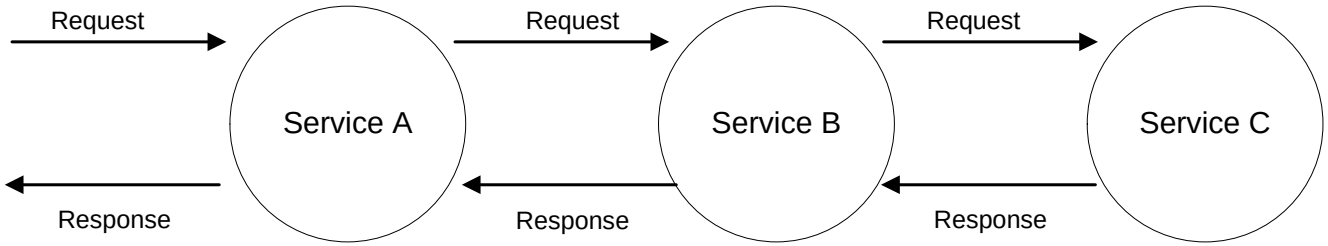
Service testing – 4 focus areas

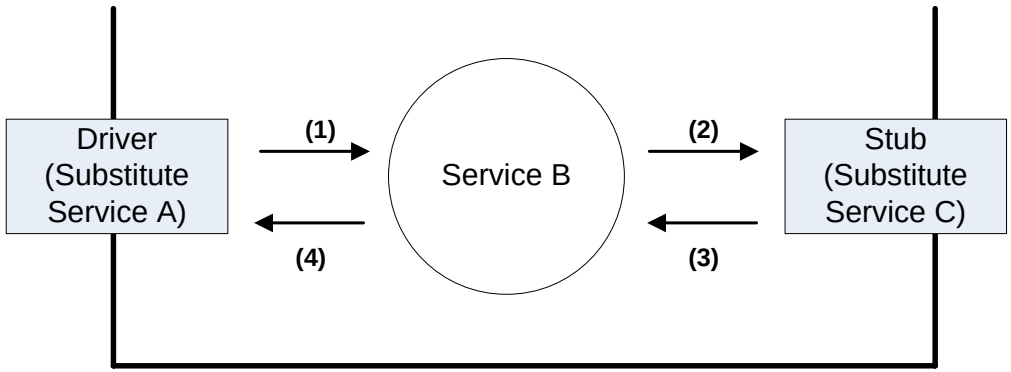
- To fully utilize the possible benefit of SOA architecture, every service consumer cannot test all functionality of the service provider.

- 4 primary focus areas: Functional, Performance, Interoperability, and Security
- **Functional** testing provides the ability to verify the proper behavior of services and build regression test suites to automate testing and baseline expected behavior
- **Performance** testing to determine throughput and capacity statistics of the service
- **Interoperability** testing to measure the design characteristics of a service as well as the runtime adherence to standards and best-practices. (testing against technical design specifications)
- **Security** testing (data integrity, data privacy, and access control)

SOA test harness

A lot of services do not have a user interface





Test harness

Some commercial SOA testing tools

Company name	**Product name SOA test harness**
GreenHat	GH Tester
IBM Software group	Rational Tester for SOA Quality
iTKO	Lisa
Parasoft	SOAtest
Crosscheck Networks	SOAPSonar
Hewlett-Packard	HP Service Test
SmartBear	soapUI (open source)

Risk with tools (from Hans)

- Unrealistic expectations
- Underestimating time, cost and effort to introduce (training, coaching, help, ...)
- Underestimating time and cost for follow-up, change of processes, etc.
- Underestimating need for maintenance of tools and the assets maintained or generated by tools
- Over-reliance on the tool
- Defects in the tool, and fighting against them
- Neglecting version control of test assets
- Neglecting relationships and interoperability between tools
- Tool vendor problems (out of business, retiring the tool)
- Sometimes too much trust into the tool instead of intelligent choices.
- Poor support
- Platform problems

Introduction of tools

- Define your process – then look for tool support.
- Remember that all tools will require resources with skills.
- Suggested sequence of tool introduction:

Book:

1. Incident management
2. Configuration management
3. Test planning
4. Test execution
5. Test specification

My suggestion:

1. Configuration management
2. Incident management
3. Test planning
4. Test specification
5. Test execution

Tool selection process

- Which testing tasks needs tool support
- Tool requirement specification
- Market research (long list of possible tools and vendors)
- Tool demos,
- Follow up references

- Create short list
- Further studies of short list candidates, if demonstration license available – try the product yourself
- Make a business case as basis for following up costs and benefits
- Review results and select

Selection criteria

- Learning curve for users (e.g. new method)
- Ease of integration with current environment
- Ease of integration with other tools
- How does the product fit in with the test object (not applicable for all tools)
- Platform (db, operating system, browsers)
- Vendors market position, customer support, reliability
- License and maintenance

Weigh the criteria and weigh how the tools meet them

Tool introduction

- Run a pilot – a pilot should result in a list of necessary tool implementation tasks *
- Evaluate results – what is to gain?
- Adapt process
- Train the users (Just in time)
- Stepwise introduction
- **Tool support is an ongoing task**

- Evaluate the benefit and calculate TOC (Total Cost of Ownership). after 1-2 years
- The cost of training and support must be included when you do calculate TOC

* Rules for usage, adaption of test procedure, naming standards, workflow, training program,

References

- Kaner, Bach, Petticord :Lessons learned in software testing, Wiley 2002
- Gartner:Magic quadrant for integrated software quality suites, ID number G00208975
- Dorothy Graham: Successful test automation, Tutorial Software 2011 (Dataforeningen)
- Software Test Automation, Mark Fewster & Dorothy Graham, Addison Wesley, 1999
- http://www.oasis-open.org/
- www.soatesting.com
- http://www.thbs.com/pdfs/SOA_Test_Methodology.pdf
- List of tools: http://www.softwareqatest.com/qatweb1.html#LOAD
- http://www.usefulusability.com/24-usability-testing-tools/#Paper
- http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks
- http://www.smashingmagazine.com/2011/08/07/a-dozen-cross-browser-testing-tools/
- http://www.satisfice.com/blog/ : testing blog of James Bach