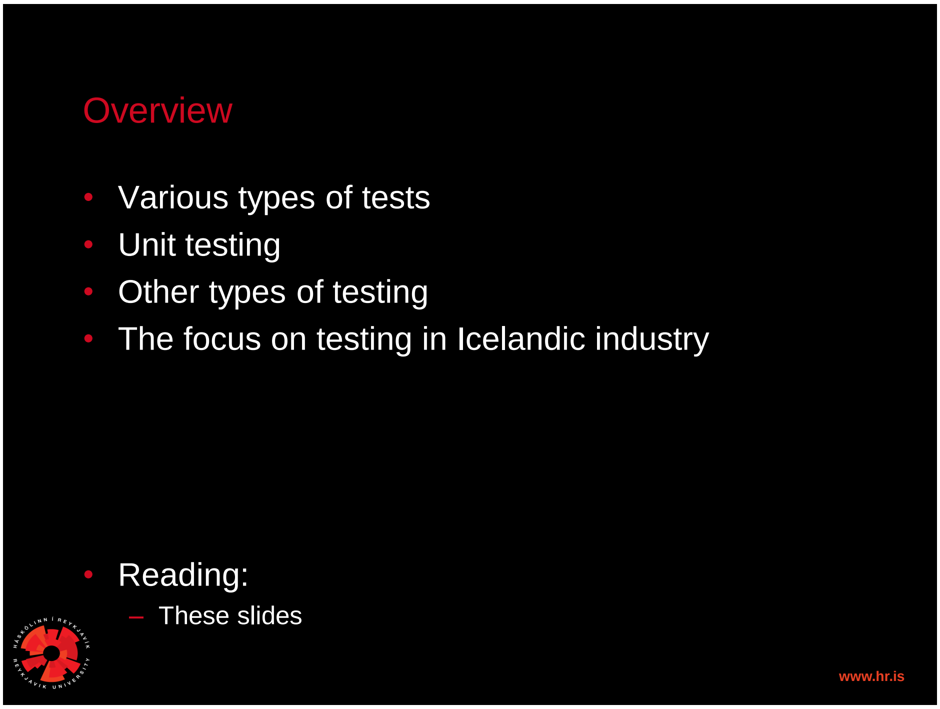
HÁSKÓLINN Í REYKJAVÍK
REYKJAVIK UNIVERSITY

Software Testing

Department of Computer Science | Software Analysis &Design
Marta Kristín Lárusdóttir, Associate professor
28. October, 2019

www.hr.is



Overview

- Various types of tests
- Unit testing
- Other types of testing
- The focus on testing in Icelandic industry

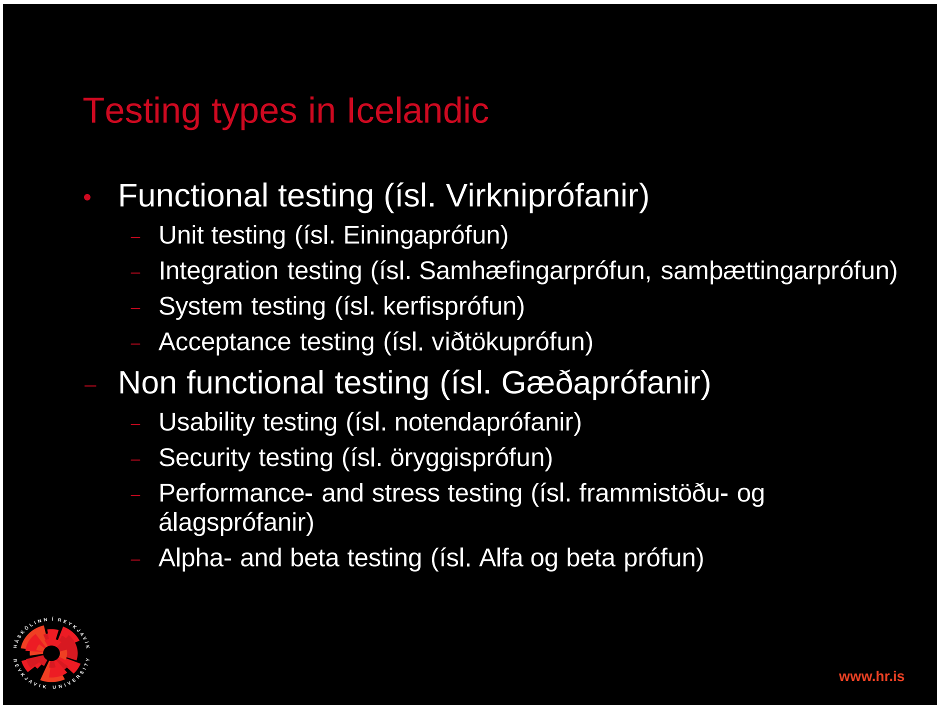
- Reading:
  - These slides

www.hr.is

Goals of testing – why test?

- Testing is the process of examining the system,
  - and its subsystems and components
- to determine its operational characteristics and whether it contains any defects
  - Verify that the system does what it was supposed to do
  - Find bugs
  -
- One must know the requirements
  - what the system should entail and accomplish
  - to do any kind of testing

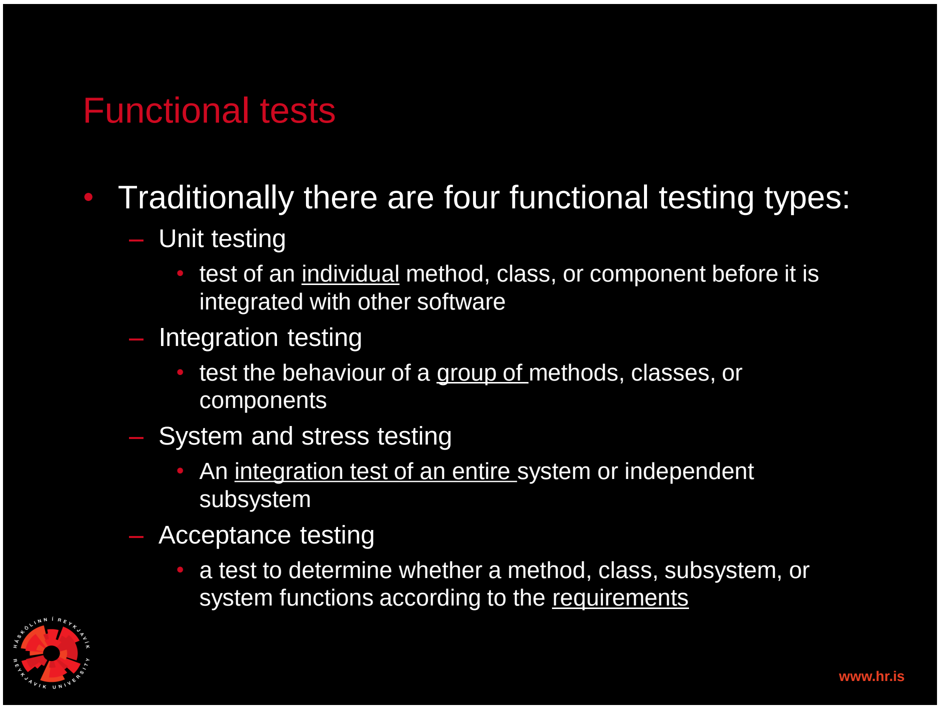
Various Types of Tests



Testing types in Icelandic

- Functional testing (ísl. Virkniprófanir)
  - Unit testing (ísl. Einingaprófun)
  - Integration testing (ísl. Samhæfingarprófun, samþættingarprófun)
  - System testing (ísl. kerfisprófun)
  - Acceptance testing (ísl. viðtökuprófun)
- Non functional testing (ísl. Gæðaprófanir)
  - Usability testing (ísl. notendaprófanir)
  - Security testing (ísl. öryggisprófun)
  - Performance- and stress testing (ísl. frammistöðu- og álagsprófanir)
  - Alpha- and beta testing (ísl. Alfa og beta prófun)

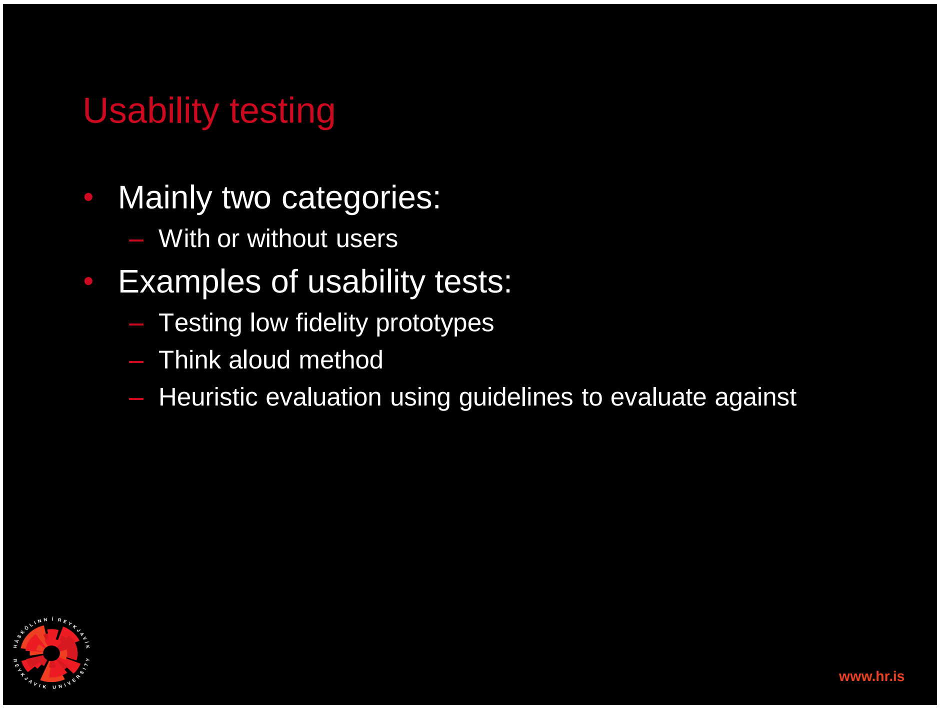
www.hr.is



Functional tests

- Traditionally there are four functional testing types:
  - Unit testing
    - test of an individual method, class, or component before it is integrated with other software
  - Integration testing
    - test the behaviour of a group of methods, classes, or components
  - System and stress testing
    - An integration test of an entire system or independent subsystem
  - Acceptance testing
    - a test to determine whether a method, class, subsystem, or system functions according to the requirements

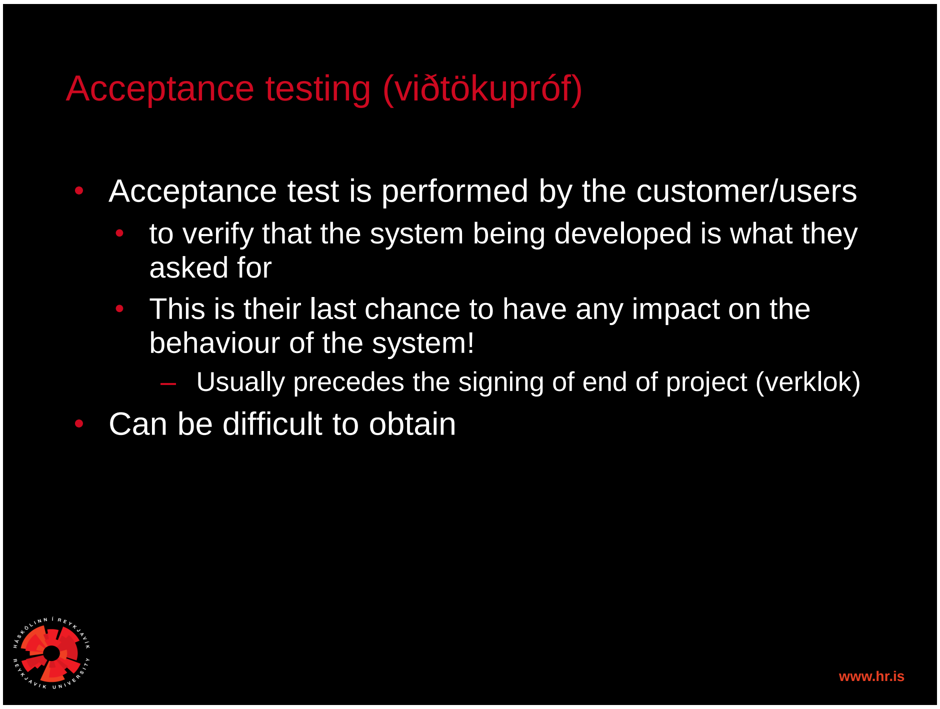
www.hr.is



Usability testing

- Mainly two categories:
  - With or without users
- Examples of usability tests:
  - Testing low fidelity prototypes
  - Think aloud method
  - Heuristic evaluation using guidelines to evaluate against

www.hr.is



Acceptance testing (viðtökupróf)

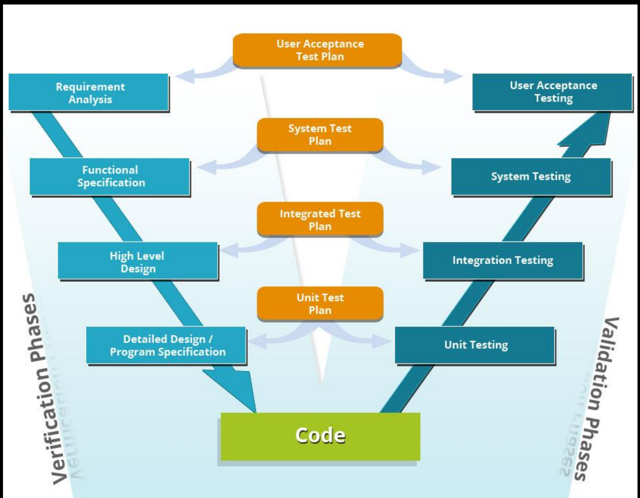
- Acceptance test is performed by the customer/users
  - to verify that the system being developed is what they asked for
  - This is their last chance to have any impact on the behaviour of the system!
    - Usually precedes the signing of end of project (verklok)
- Can be difficult to obtain

www.hr.is

Other tests

- Alpha- and beta tests
  - used to test the software in real-life circumstances, with a disclaimer that the software is buggy and the user should be prepared for data loss
    - alpha test: the system barely works!
    - beta test: Fairly sure it does what it should do, but we're not done
- Performance testing
  - Is the process of testing to determine the performance and/or measuring the behavior of a component or system with increasing load
- Stress tests
  - often, a program is written that issues multiple requests to the system that is being tested (thousands/millions)
- Security tests
  - the system is tested for security holes

The Various Testing Types



User Acceptance Test Plan

Requirement Analysis

User Acceptance Testing

System Test Plan

Functional Specification

System Testing

Integrated Test Plan

High Level Design

Integration Testing

Unit Test Plan

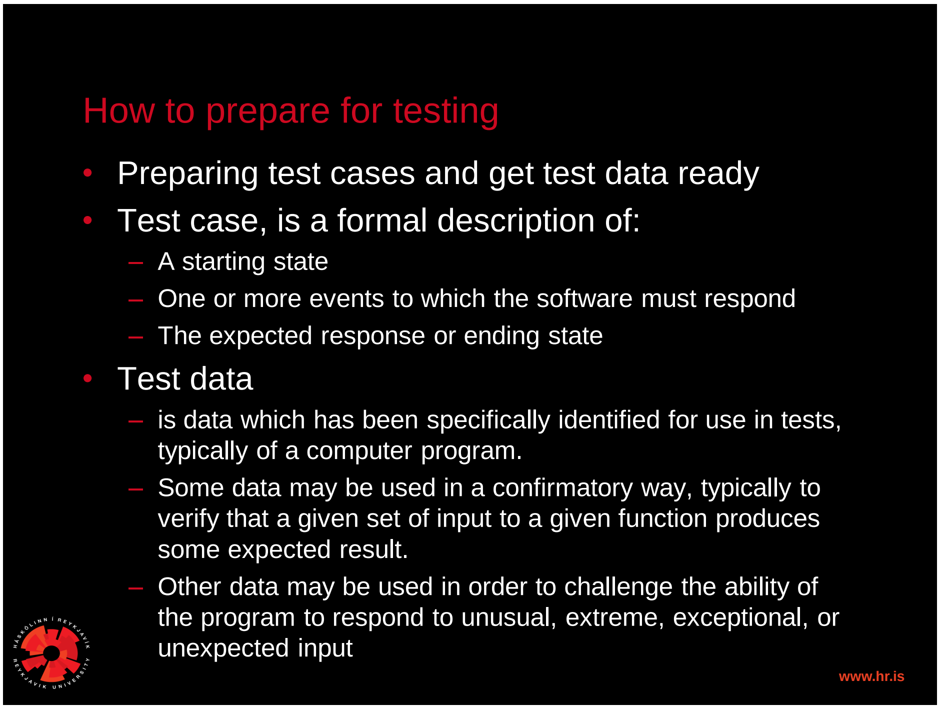
Detailed Design / Program Specification

Unit Testing

Code

Verification Phases

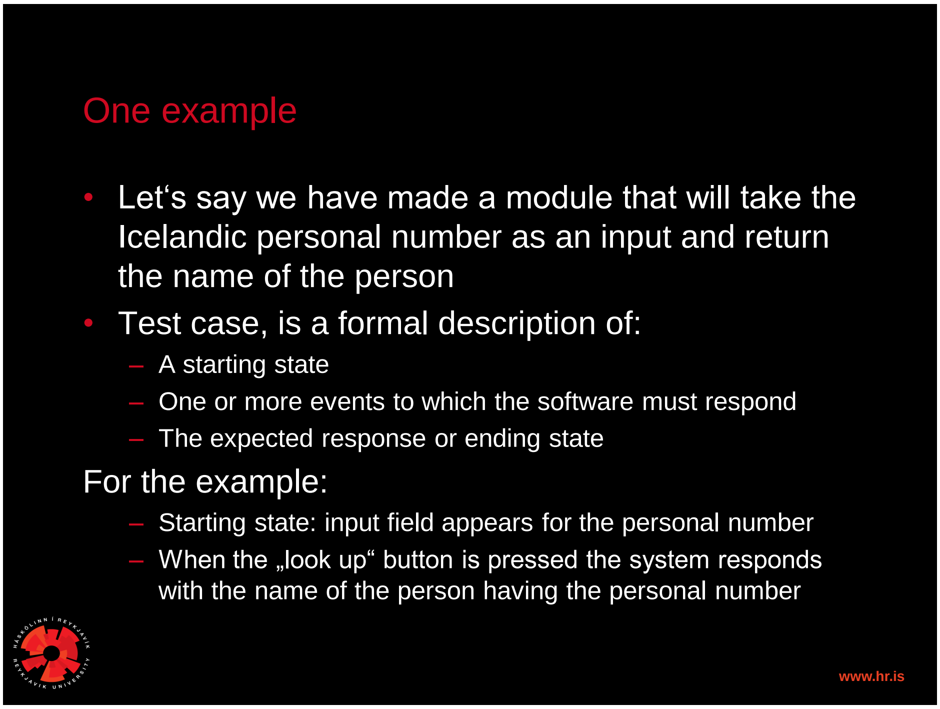
Validation Phases



How to prepare for testing

- Preparing test cases and get test data ready
- Test case, is a formal description of:
  - A starting state
  - One or more events to which the software must respond
  - The expected response or ending state
- Test data
  - is data which has been specifically identified for use in tests, typically of a computer program.
  - Some data may be used in a confirmatory way, typically to verify that a given set of input to a given function produces some expected result.
  - Other data may be used in order to challenge the ability of the program to respond to unusual, extreme, exceptional, or unexpected input

www.hr.is



One example

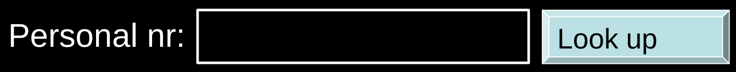
- Let‘s say we have made a module that will take the Icelandic personal number as an input and return the name of the person
- Test case, is a formal description of:
  - A starting state
  - One or more events to which the software must respond
  - The expected response or ending state

For the example:

  - Starting state: input field appears for the personal number
  - When the „look up“ button is pressed the system responds with the name of the person having the personal number

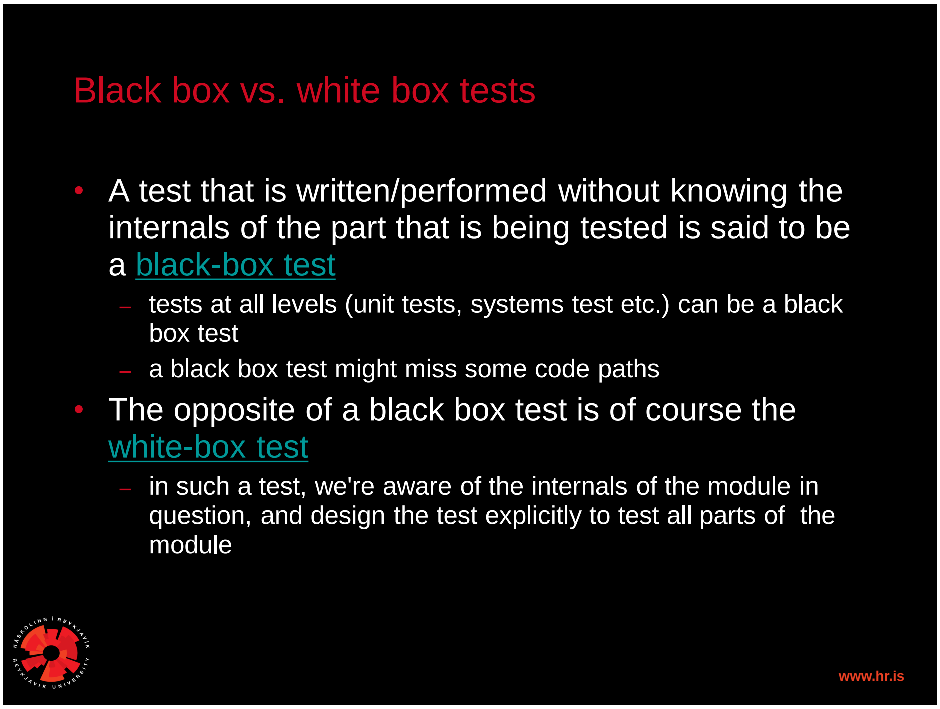
www.hr.is

What Would You Test Here?


Personal nr:

The example – cont.

- The test data – happy path
  - Some data may be used in a confirmatory way, typically to verify that a given set of input to a given function produces some expected result.
  - We could input: 0806032269 – should return: Marta Kristín Lárusdóttir
- When something goes wrong
  - Other data may be used in order to challenge the ability of the program to respond to unusual, extreme, exceptional, or unexpected input
  - We could input: 0000000000 – Should return: no record was found



Black box vs. white box tests

- A test that is written/performed without knowing the internals of the part that is being tested is said to be a black-box test
  - tests at all levels (unit tests, systems test etc.) can be a black box test
  - a black box test might miss some code paths
- The opposite of a black box test is of course the white-box test
  - in such a test, we're aware of the internals of the module in question, and design the test explicitly to test all parts of the module

www.hr.is

Preparing

- Preparing test cases and test data takes time and can be very complex

www.hr.is

Unit testing

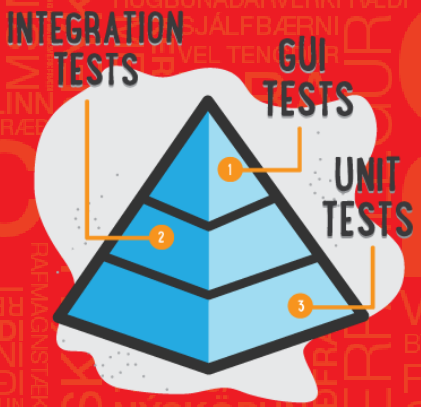


ILLUSTRATION BY SEGUE TECHNOLOGIES

www.hr.is

Unit Tests

- Unit testing
  - – test of an individual method, class, or component before it is integrated with other software
- Test driven development
  - Write tests first, code later
- Unit tests could be either blackbox or whitebox
  - Blackbox – you are testing, if the unit returns the right output given a particular input
  - Whitebox – you know the steps and you are checking, if the state is right in each stage of the unit for all the paths through the unit

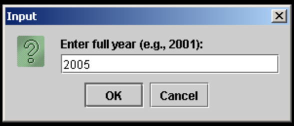
www.hr.is

Unit Testing - Stepwise Refinement

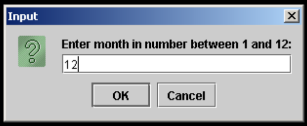
- Method abstraction can be used to simplify design and implementation of programs.
- If you are writing a big and complicated program
  - It is easier to brake the program down in small units, because smaller units are usually simpler then a whole program
  - This is called “divide and conquer” or “stepwise refinement”
  - Then you can take each unit and break it into smaller units, etc.
- Lets look at the program: PrintCalendar

A Program - PrintCalendar

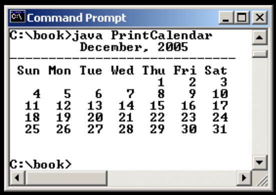
We will use the PrintCalendar example to explain.



Enter full year (e.g., 2001): 2005



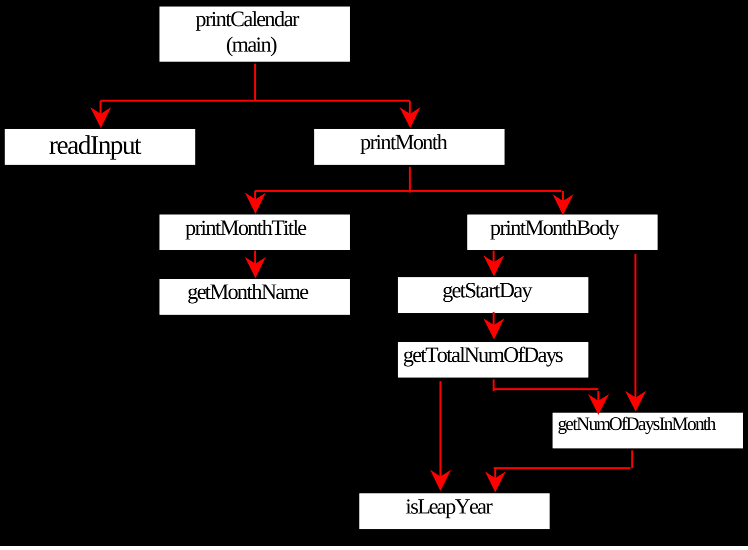
Enter month in number between 1 and 12: 12



```
C:\book>java PrintCalendar
         December, 2005
-----------------------------
 Sun Mon Tue Wed Thu Fri Sat
                   1   2   3
   4   5   6   7   8   9  10
  11  12  13  14  15  16  17
  18  19  20  21  22  23  24
  25  26  27  28  29  30  31

C:\book>
```

Overview Diagram - Design Diagram



printCalendar
(main)

readInput

printMonth

printMonthTitle

printMonthBody

getMonthName

getStartDay

getTotalNumOfDays

getNumOfDaysInMonth

isLeapYear

Top-Down Implementation

- When using Top-down Implementation
  1. you start by developing the units that are at the highest level in the Overview Diagram (printCalendar (main))
  2. then you make the ones on the next level until you have reached the units in the bottom level (isLeapYear).
- You can program some stubs for the functions that are not developed yet to make testing easier
  - For example the printMonth function could display year and month in the stub.

Bottom-Up Implementation

- While implementing Bottom-up
  1. first you implement the functions on the lowest level on the Overview Diagram
  2. then on the level above until you have reached the top level
- When implementing like this
  - you need to make programs to test that are using the programs below

An example of a Unit Test

```
// one way to do unit testing is to do them before you implement

int main()
      //test program for Print Calendar
           assert(isLeapYear(2000));
                //should return true, otherwise there is an error
           assert(isLeapYear(1996));
               //and there is not a LeapYear
           assert(!isLeapYear(1999));
               //assert(getNumOfDayInMonth(here you probably need a variable for year and month)
           assert(getNumOfDayInMonth(1,2000)==31);
               //and more
           assert(getNumOfDayInMonth(2,2000)==29);
           assert(getNumOfDayInMonth(2,2001)==28);
               //etc.
```

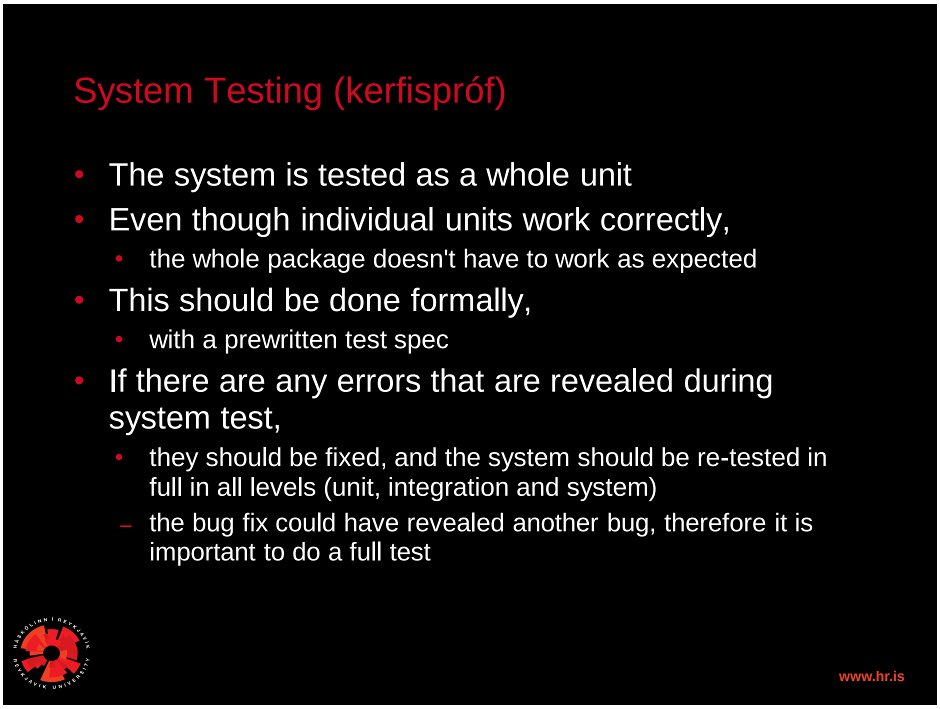
Other types of Testing

www.hr.is

Integration Testing (samhæfingarpróf)

- System Integration Testing
  - or just Integration test tests
  - if individual units (classes/modules) work correctly together
- Is performed after the modules pass their unit tests
  - but before the system test

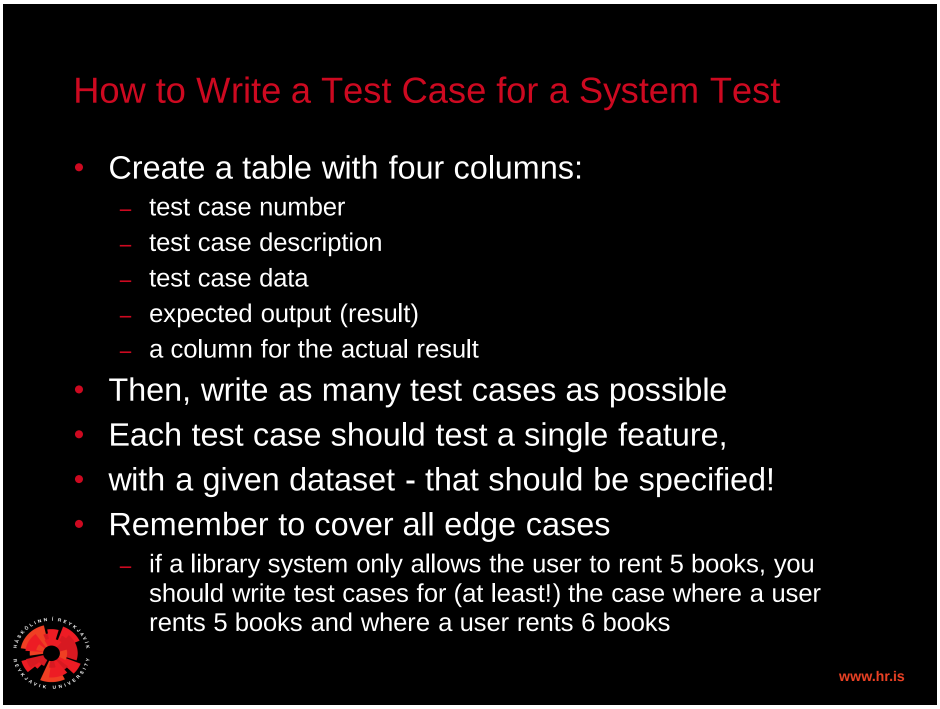
www.hr.is



System Testing (kerfispróf)

- The system is tested as a whole unit
- Even though individual units work correctly,
  - the whole package doesn't have to work as expected
- This should be done formally,
  - with a prewritten test spec
- If there are any errors that are revealed during system test,
  - they should be fixed, and the system should be re-tested in full in all levels (unit, integration and system)
  - the bug fix could have revealed another bug, therefore it is important to do a full test

www.hr.is

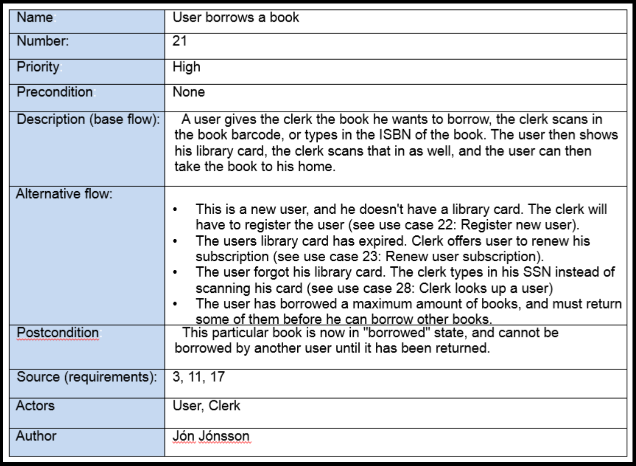


How to Write a Test Case for a System Test

- Create a table with four columns:
  - test case number
  - test case description
  - test case data
  - expected output (result)
  - a column for the actual result
- Then, write as many test cases as possible
- Each test case should test a single feature,
- with a given dataset - that should be specified!
- Remember to cover all edge cases
  - if a library system only allows the user to rent 5 books, you should write test cases for (at least!) the case where a user rents 5 books and where a user rents 6 books

www.hr.is

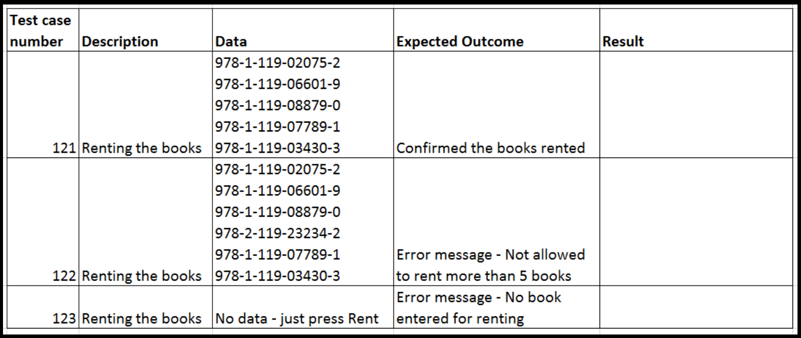
Use Case – An example



Name	User borrows a book
Number:	21
Priority	High
Precondition	None
Description (base flow):	A user gives the clerk the book he wants to borrow, the clerk scans in the book barcode, or types in the ISBN of the book. The user then shows his library card, the clerk scans that in as well, and the user can then take the book to his home.
Alternative flow:	• This is a new user, and he doesn't have a library card. The clerk will have to register the user (see use case 22: Register new user). • The users library card has expired. Clerk offers user to renew his subscription (see use case 23: Renew user subscription). • The user forgot his library card. The clerk types in his SSN instead of scanning his card (see use case 28: Clerk looks up a user) • The user has borrowed a maximum amount of books, and must return some of them before he can borrow other books.
Postcondition	This particular book is now in "borrowed" state, and cannot be borrowed by another user until it has been returned.
Source (requirements):	3, 11, 17
Actors	User, Clerk
Author	Jón Jónsson

www.hr.is

Test Case Descriptions



Test case number	Description	Data	Expected Outcome	Result
121	Renting the books	978-1-119-02075-2 978-1-119-06601-9 978-1-119-08879-0 978-1-119-07789-1 978-1-119-03430-3	Confirmed the books rented	
122	Renting the books	978-1-119-02075-2 978-1-119-06601-9 978-1-119-08879-0 978-2-119-23234-2 978-1-119-07789-1 978-1-119-03430-3	Error message - Not allowed to rent more than 5 books	
123	Renting the books	No data - just press Rent	Error message - No book entered for renting	

- You can use the requirements’ list and use cases
  - one use case can mean one or many test cases

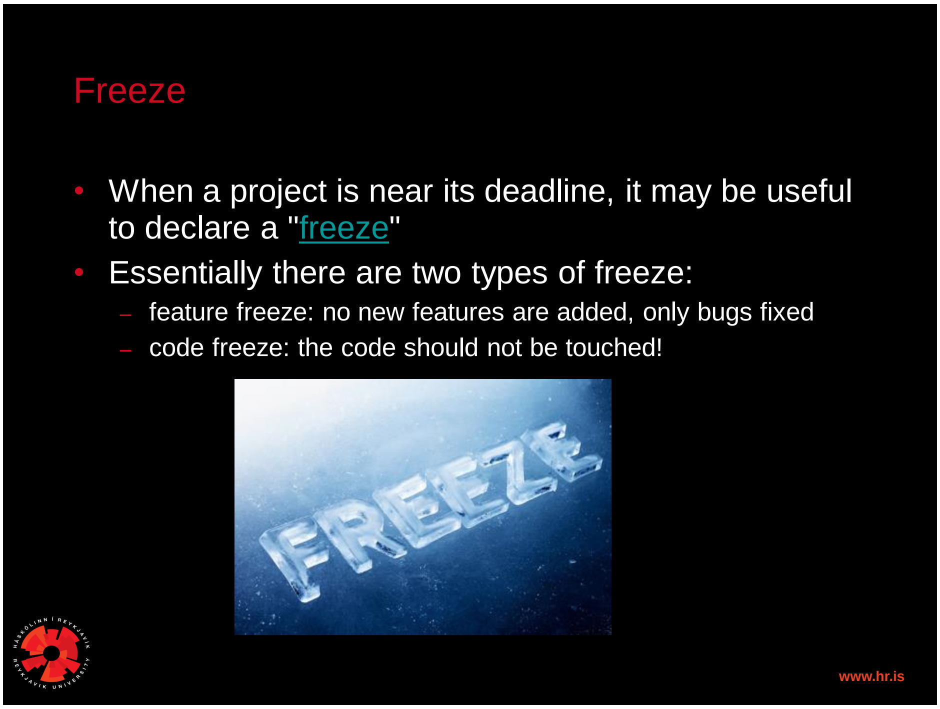
www.hr.is

Functional testing for MMDS

EJS

The Lessons Learned from that Experience

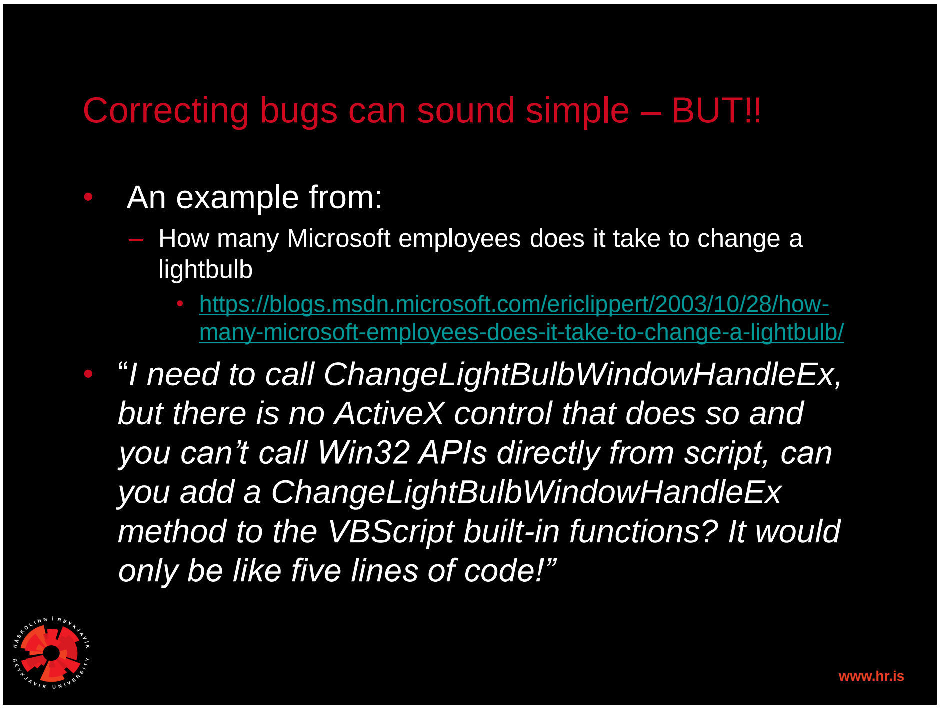
- Even though the processes were very well defined
  - ISO 9001 accreditation
- And we consulted with domain experts
  - We failed to provide the users with a system they needed
- Try to test with users as soon as possible
  - To know if you have understood the requirements
- Design the right system
  - Means we need to check with users, if they need the functionality provided
- Designing the system right
  - The system you have designed is without bugs and easy to use



Freeze

- When a project is near its deadline, it may be useful to declare a "freeze"
- Essentially there are two types of freeze:
  - feature freeze: no new features are added, only bugs fixed
  - code freeze: the code should not be touched!

www.hr.is

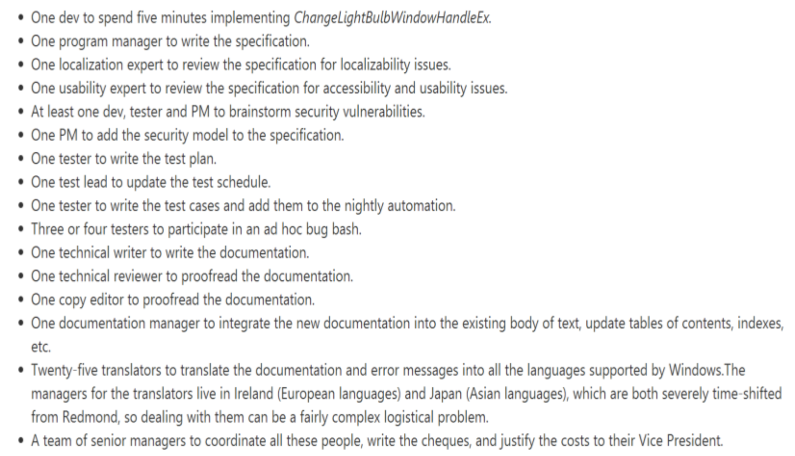


Correcting bugs can sound simple – BUT!!

- An example from:
  - How many Microsoft employees does it take to change a lightbulb
    - https://blogs.msdn.microsoft.com/ericlippert/2003/10/28/how-many-microsoft-employees-does-it-take-to-change-a-lightbulb/
- *“I need to call ChangeLightBulbWindowHandleEx, but there is no ActiveX control that does so and you can’t call Win32 APIs directly from script, can you add a ChangeLightBulbWindowHandleEx method to the VBScript built-in functions? It would only be like five lines of code!”*

www.hr.is

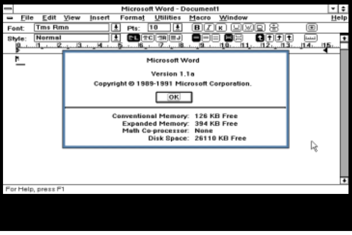
How many people involved?



- One dev to spend five minutes implementing *ChangeLightBulbWindowHandleEx*.
- One program manager to write the specification.
- One localization expert to review the specification for localizability issues.
- One usability expert to review the specification for accessibility and usability issues.
- At least one dev, tester and PM to brainstorm security vulnerabilities.
- One PM to add the security model to the specification.
- One tester to write the test plan.
- One test lead to update the test schedule.
- One tester to write the test cases and add them to the nightly automation.
- Three or four testers to participate in an ad hoc bug bash.
- One technical writer to write the documentation.
- One technical reviewer to proofread the documentation.
- One copy editor to proofread the documentation.
- One documentation manager to integrate the new documentation into the existing body of text, update tables of contents, indexes, etc.
- Twenty-five translators to translate the documentation and error messages into all the languages supported by Windows.The managers for the translators live in Ireland (European languages) and Japan (Asian languages), which are both severely time-shifted from Redmond, so dealing with them can be a fairly complex logistical problem.
- A team of senior managers to coordinate all these people, write the cheques, and justify the costs to their Vice President.

Should we fix all bugs?

- When deciding if a bug should be fixed,
  - we must consider if the bug fix will have an impact on other features
  - bug fixes (or new feature) may uncover other (more serious) bugs
- According to Joel Spolsky, we should fix bugs before we write new code/features (see item no. 5)
  - Very good example for Microsoft Word



Microsoft Word - Document1

Microsoft Word

Version 1.1a

Copyright © 1989-1991 Microsoft Corporation.

OK

Conventional Memory: 126 KB Free
Expanded Memory: 394 KB Free
Math Co-processor: None
Disk Space: 26110 KB Free

For Help, press F1

The Focus on Usability Testing in Scrum Projects in Iceland

Paper presented at the World Computer Congress in Australin 2010

www.hr.is

The Authors

Marta Lárusdóttir

Emma Rún Bjarnadóttir

Jan Gulliksen

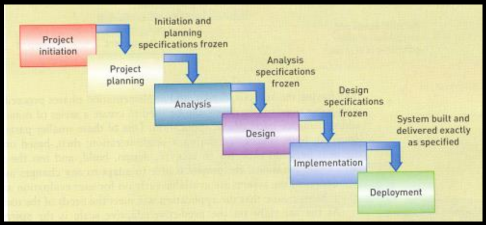
www.hr.is

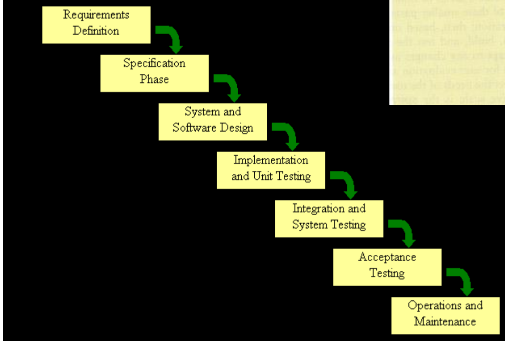
The Motivation for the Study

- Scrum has become popular in resent year in Iceland
  - 1/3 of respondent of a survey for software developers in Iceland used Scrum in spring 2009
  - The popularity is growing
- The motivation
  - What testing is conducted?
  - How often is it conducted?
  - How easy is it to do testing in Scrum projects?
  - What are the hindrances of conducting testing?
  - What is the difference of usability and accessability testing?

www.hr.is

Waterfall model


Project initiation
Initiation and planning specifications frozen
Project planning
Analysis
Analysis specifications frozen
Design
Design specifications frozen
Implementation
System built and delivered exactly as specified
Deployment



Requirements Definition

Specification Phase

System and Software Design

Implementation and Unit Testing

Integration and System Testing

Acceptance Testing

Operations and Maintenance

www.hr.is

Scrum

- Very popular today

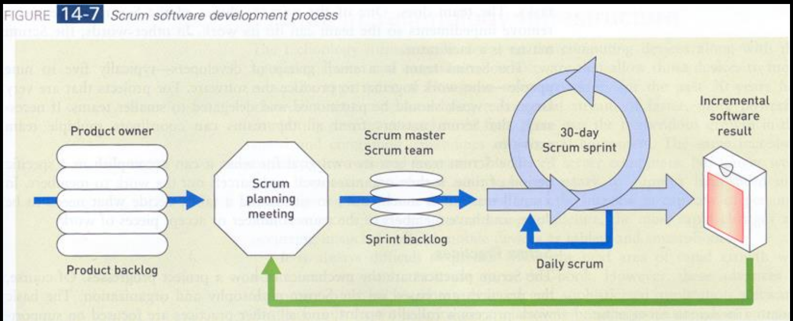
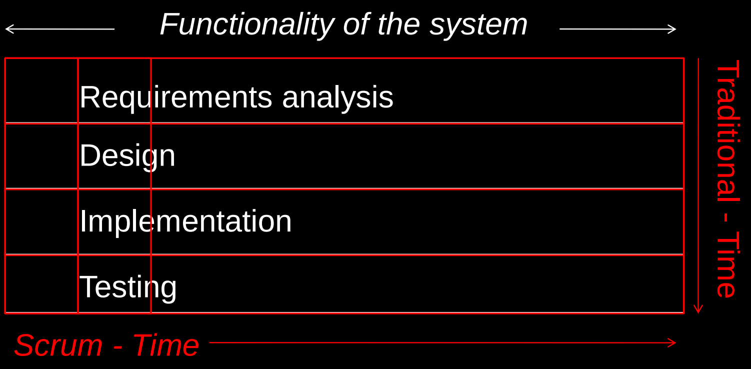


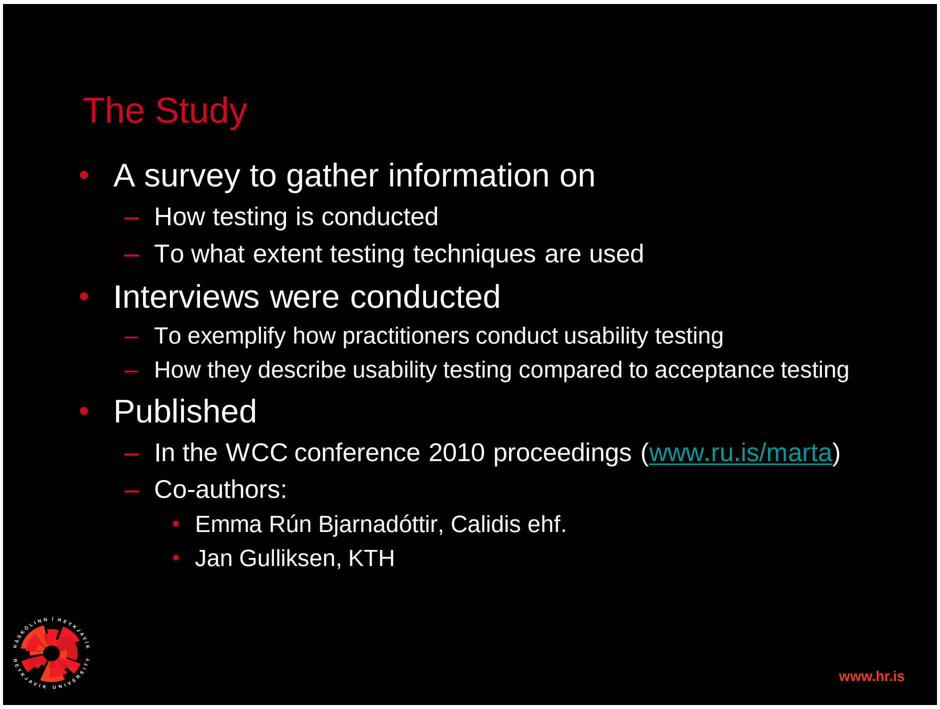
FIGURE 14-7 Scrum software development process

What is Scrum?

- It is a Software development process



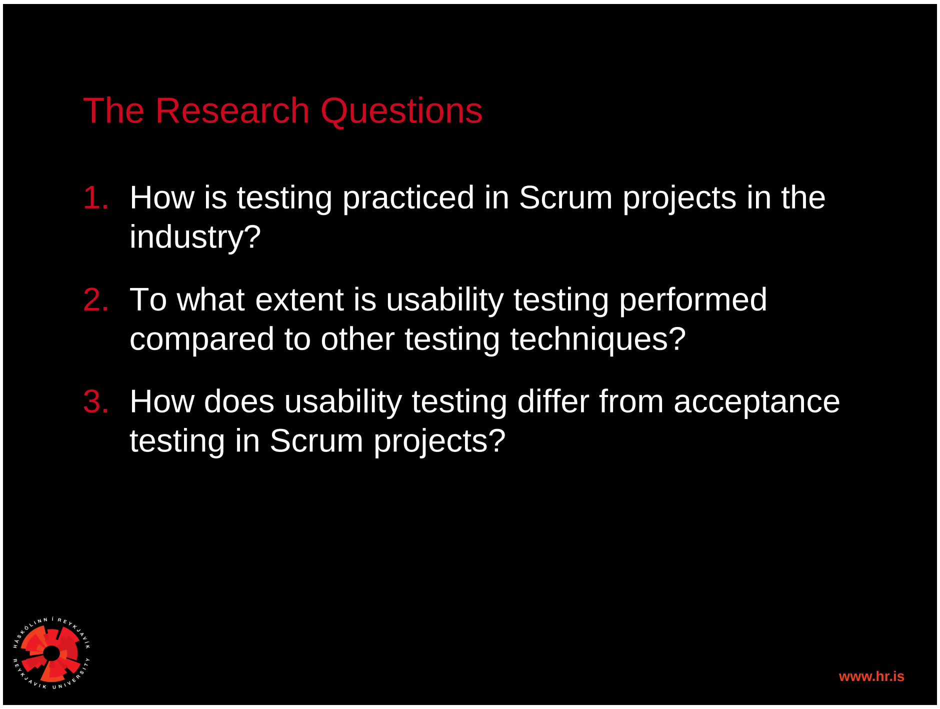
- Scrum roles: Scrum Master, Product Owner, team



The Study

- A survey to gather information on
  - How testing is conducted
  - To what extent testing techniques are used
- Interviews were conducted
  - To exemplify how practitioners conduct usability testing
  - How they describe usability testing compared to acceptance testing
- Published
  - In the WCC conference 2010 proceedings (www.ru.is/marta)
  - Co-authors:
    - Emma Rún Bjarnadóttir, Calidis ehf.
    - Jan Gulliksen, KTH

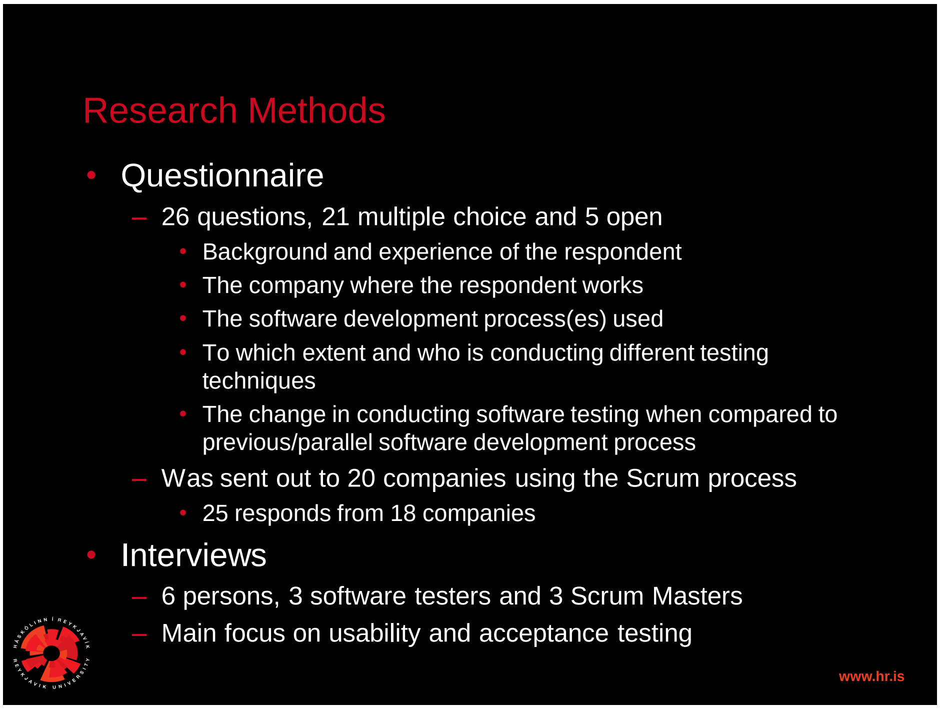
www.hr.is



The Research Questions

1. How is testing practiced in Scrum projects in the industry?
2. To what extent is usability testing performed compared to other testing techniques?
3. How does usability testing differ from acceptance testing in Scrum projects?

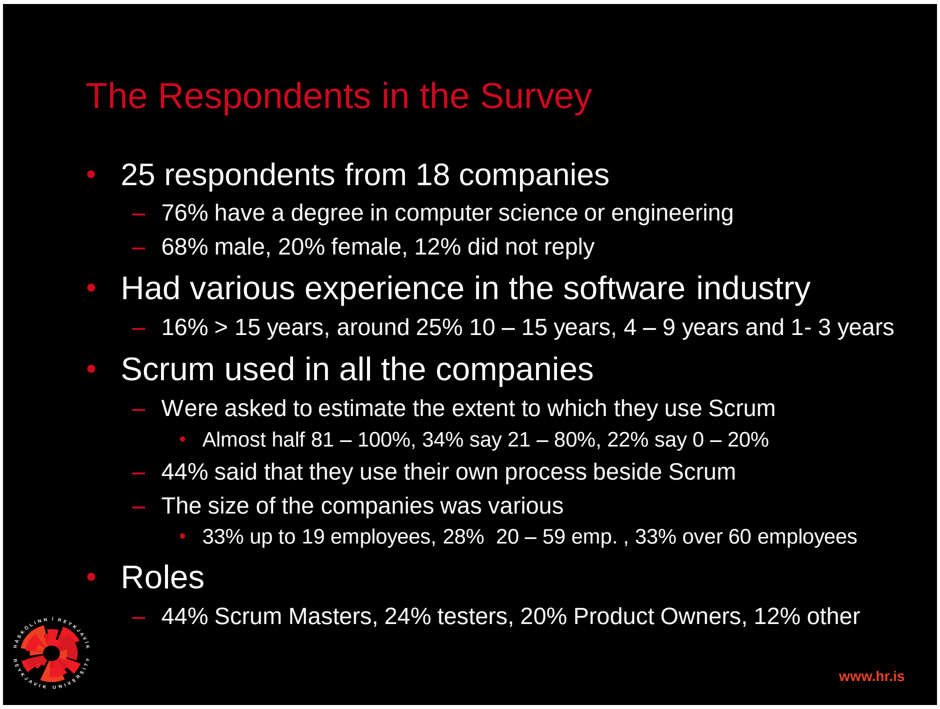
www.hr.is



Research Methods

- Questionnaire
  - 26 questions, 21 multiple choice and 5 open
    - Background and experience of the respondent
    - The company where the respondent works
    - The software development process(es) used
    - To which extent and who is conducting different testing techniques
    - The change in conducting software testing when compared to previous/parallel software development process
  - Was sent out to 20 companies using the Scrum process
    - 25 responds from 18 companies
- Interviews
  - 6 persons, 3 software testers and 3 Scrum Masters
  - Main focus on usability and acceptance testing

www.hr.is

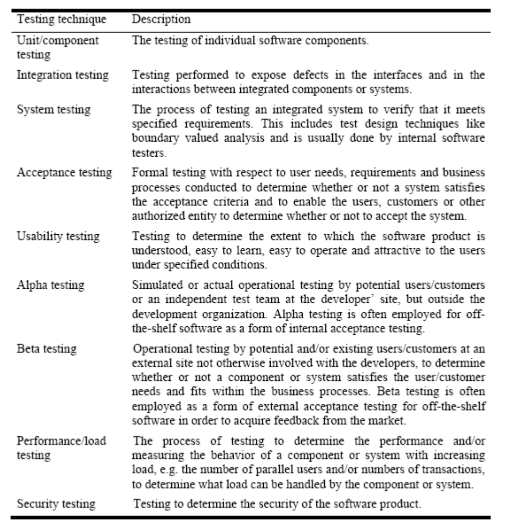


The Respondents in the Survey

- 25 respondents from 18 companies
  - 76% have a degree in computer science or engineering
  - 68% male, 20% female, 12% did not reply
- Had various experience in the software industry
  - 16% > 15 years, around 25% 10 – 15 years, 4 – 9 years and 1- 3 years
- Scrum used in all the companies
  - Were asked to estimate the extent to which they use Scrum
    - Almost half 81 – 100%, 34% say 21 – 80%, 22% say 0 – 20%
  - 44% said that they use their own process beside Scrum
  - The size of the companies was various
    - 33% up to 19 employees, 28% 20 – 59 emp. , 33% over 60 employees
- Roles
  - 44% Scrum Masters, 24% testers, 20% Product Owners, 12% other

www.hr.is

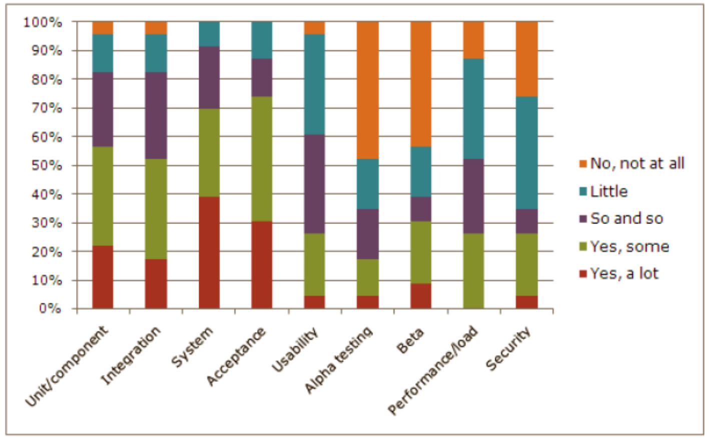
Description of Testing Techniques



Testing technique	Description
Unit/component testing	The testing of individual software components.
Integration testing	Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems.
System testing	The process of testing an integrated system to verify that it meets specified requirements. This includes test design techniques like boundary valued analysis and is usually done by internal software testers.
Acceptance testing	Formal testing with respect to user needs, requirements and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the users, customers or other authorized entity to determine whether or not to accept the system.
Usability testing	Testing to determine the extent to which the software product is understood, easy to learn, easy to operate and attractive to the users under specified conditions.
Alpha testing	Simulated or actual operational testing by potential users/customers or an independent test team at the developer' site, but outside the development organization. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing.
Beta testing	Operational testing by potential and/or existing users/customers at an external site not otherwise involved with the developers, to determine whether or not a component or system satisfies the user/customer needs and fits within the business processes. Beta testing is often employed as a form of external acceptance testing for off-the-shelf software in order to acquire feedback from the market.
Performance/load testing	The process of testing to determine the performance and/or measuring the behavior of a component or system with increasing load, e.g. the number of parallel users and/or numbers of transactions, to determine what load can be handled by the component or system.
Security testing	Testing to determine the security of the software product.

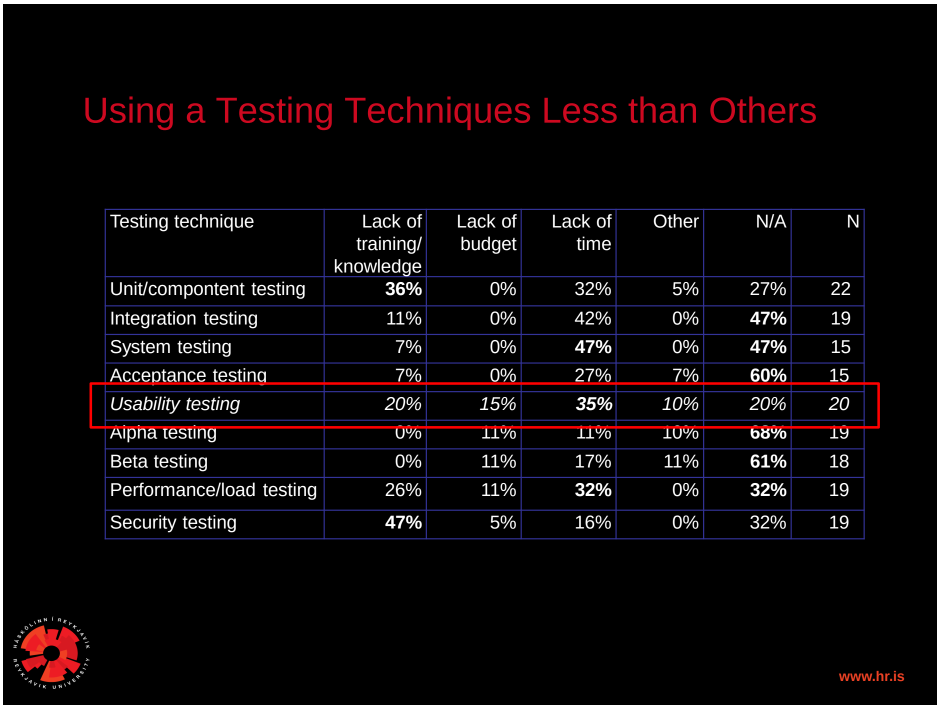
www.hr.is

To Which Extent is Testing Done?



To which extent are the following types of software testing done in Scrum (Agile) projects? Each type may be done by your Scrum (Agile) team members (company internal) and/or customers (company external), see next question.

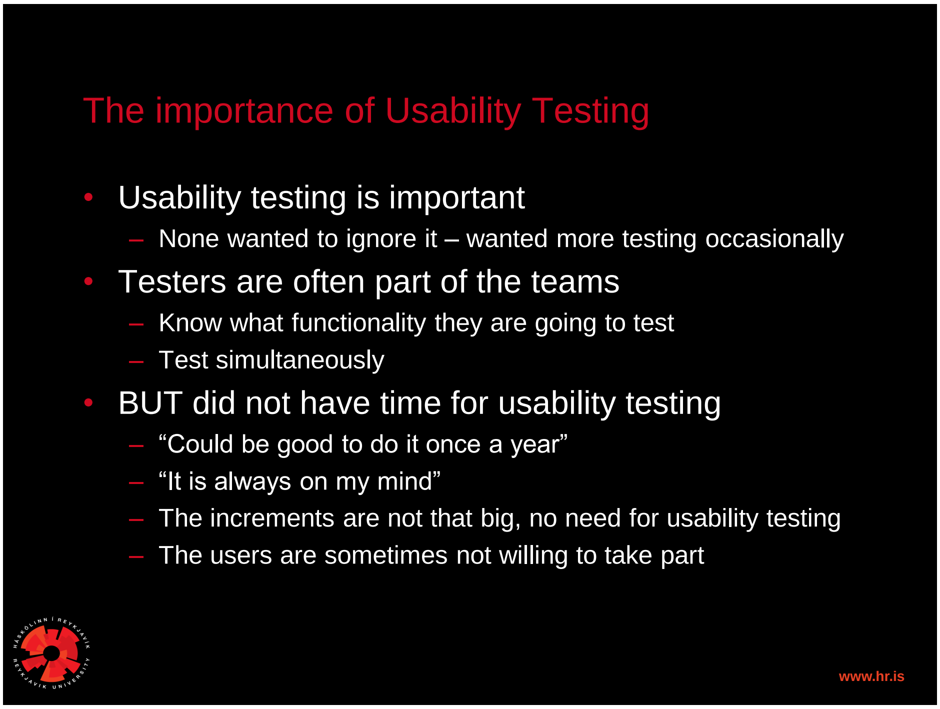
www.hr.is



Using a Testing Techniques Less than Others

Testing technique	Lack of training/ knowledge	Lack of budget	Lack of time	Other	N/A	N
Unit/compontent testing	**36%**	0%	32%	5%	27%	22
Integration testing	11%	0%	42%	0%	**47%**	19
System testing	7%	0%	**47%**	0%	**47%**	15
Acceptance testing	7%	0%	27%	7%	**60%**	15
Usability testing	*20%*	*15%*	***35%***	*10%*	*20%*	*20*
Alpha testing	0%	11%	11%	10%	**68%**	19
Beta testing	0%	11%	17%	11%	**61%**	18
Performance/load testing	26%	11%	**32%**	0%	**32%**	19
Security testing	**47%**	5%	16%	0%	32%	19

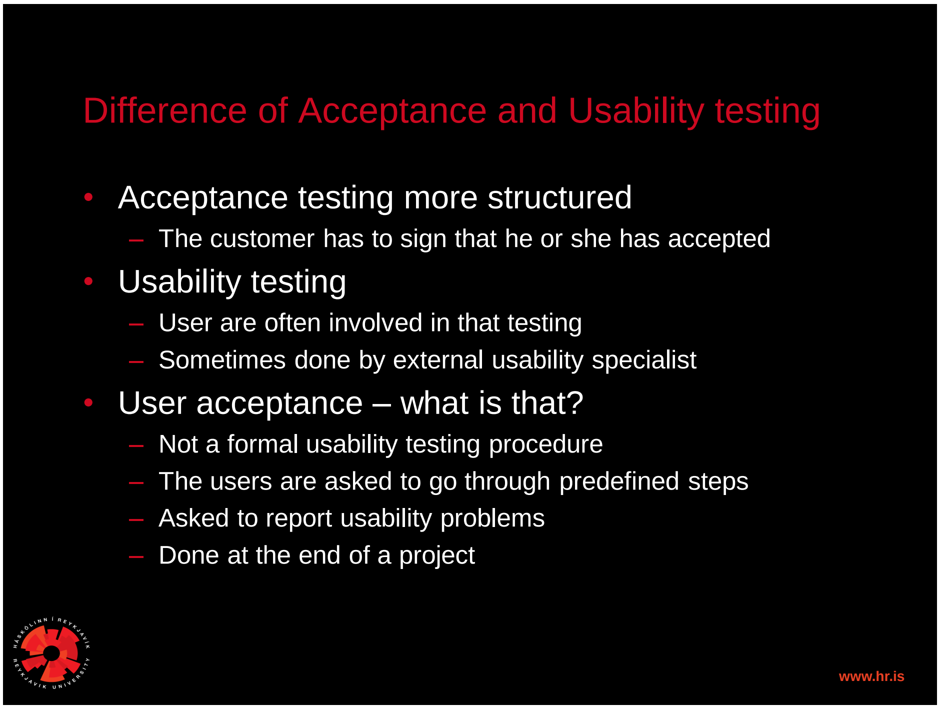
www.hr.is



The importance of Usability Testing

- Usability testing is important
  - None wanted to ignore it – wanted more testing occasionally
- Testers are often part of the teams
  - Know what functionality they are going to test
  - Test simultaneously
- BUT did not have time for usability testing
  - “Could be good to do it once a year”
  - “It is always on my mind”
  - The increments are not that big, no need for usability testing
  - The users are sometimes not willing to take part

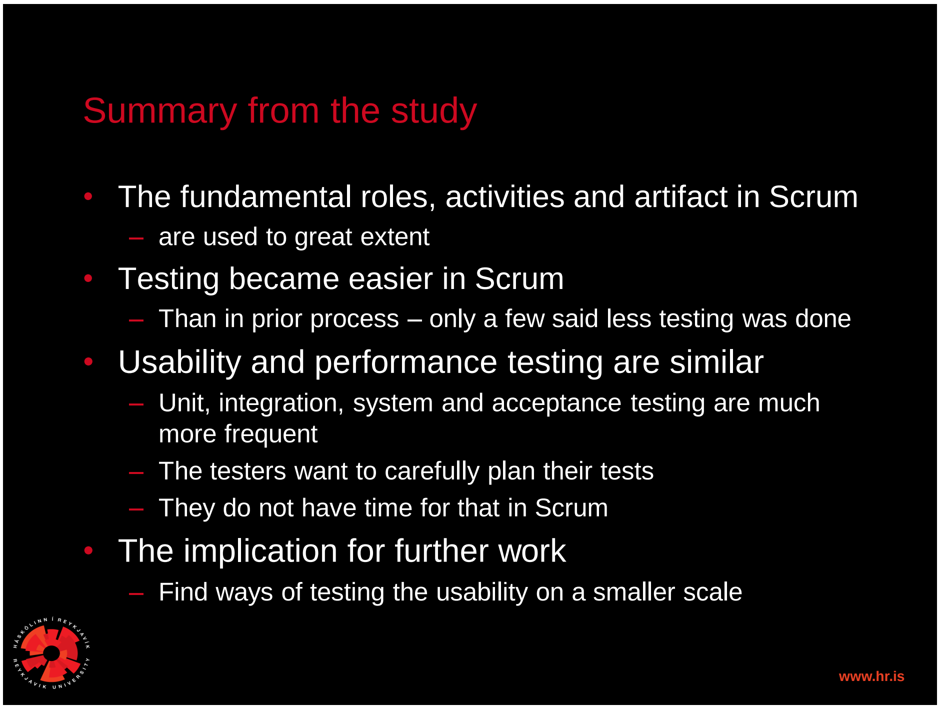
www.hr.is



Difference of Acceptance and Usability testing

- Acceptance testing more structured
  - The customer has to sign that he or she has accepted
- Usability testing
  - User are often involved in that testing
  - Sometimes done by external usability specialist
- User acceptance – what is that?
  - Not a formal usability testing procedure
  - The users are asked to go through predefined steps
  - Asked to report usability problems
  - Done at the end of a project

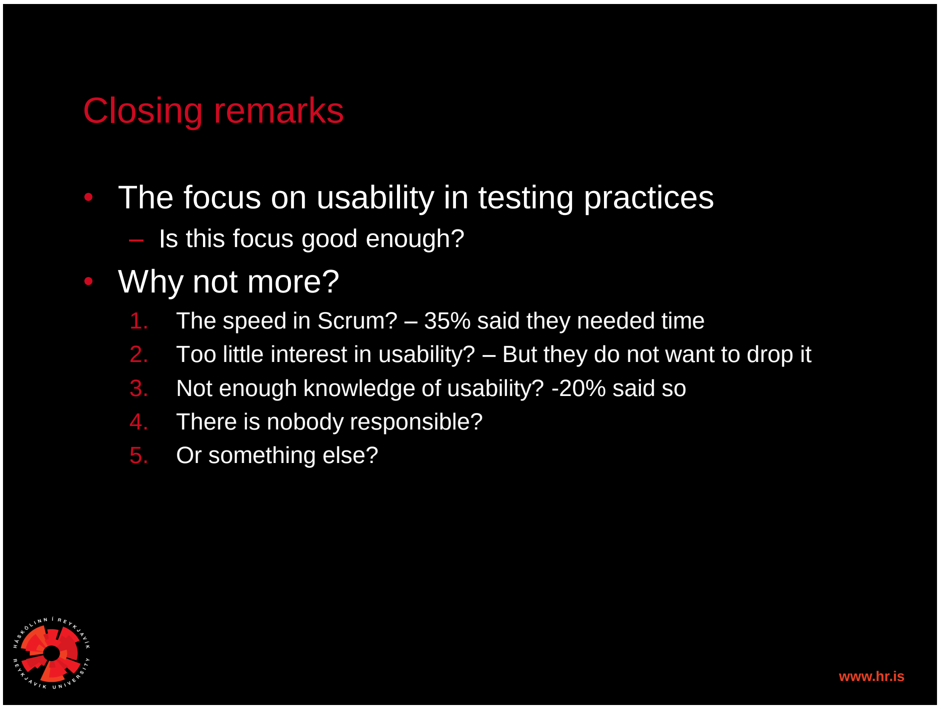
www.hr.is



Summary from the study

- The fundamental roles, activities and artifact in Scrum
  - are used to great extent
- Testing became easier in Scrum
  - Than in prior process – only a few said less testing was done
- Usability and performance testing are similar
  - Unit, integration, system and acceptance testing are much more frequent
  - The testers want to carefully plan their tests
  - They do not have time for that in Scrum
- The implication for further work
  - Find ways of testing the usability on a smaller scale

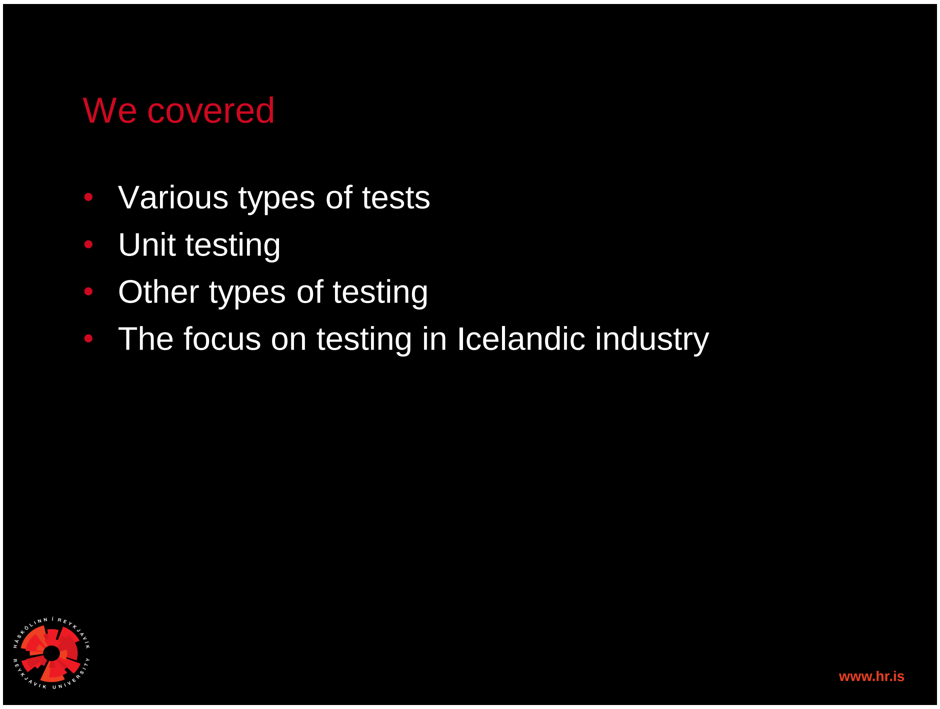
www.hr.is



Closing remarks

- The focus on usability in testing practices
  - Is this focus good enough?
- Why not more?
  1. The speed in Scrum? – 35% said they needed time
  2. Too little interest in usability? – But they do not want to drop it
  3. Not enough knowledge of usability? -20% said so
  4. There is nobody responsible?
  5. Or something else?

www.hr.is



We covered

- Various types of tests
- Unit testing
- Other types of testing
- The focus on testing in Icelandic industry

www.hr.is