

PA2516: Verification and Validation

Lecture 1: Introduction

Kai Petersen, PhD
Blekinge Institute of Technology



Teachers

- Kai Petersen (Course responsible, lectures, feedback and evaluation of exercises, seminar)
 - Contact at: kai.petersen@bth.se
 - Room: J2113
- Nauman Bin Ali (Feedback and evaluation of exercises, Seminar)
 - Contact at: nauman.ali@bth.se
 - Room: J2130

Agenda

- Give an overview of central terminology and the structure of the area
- Present course structure, learning outcomes and goals, course moments

Pre-test

- We would like to know more about you and what you have done before
- Goal: Tailor the course to your previous experience
- The pre-test will not be grade and will not have any influence on your course grade
- The test should be done individually and without any additional material (e.g. information from the internet)
- Time for doing the test: 15 Min.

Verification and Validation

- **Verification**

- **Checking of whether we are building the product right**

- **Validation**

- **Checking whether we are building the right product**

- Example

- **Hacking Wood:** If we hack someone else's wood very fast (pieces of wood/minute) then we are highly efficient (build the product right), but not effective (build the right product)
- **Developing Software:** If we develop requirements with a high velocity and introduce few defects on the way we are highly efficient (build the product right). If the software does not fulfill the needs of the customer though, we are not effective (build the right product)

Why V&V?

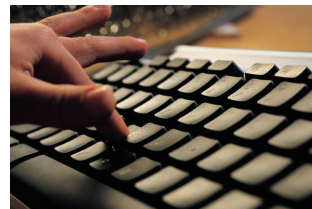
Requirements



Design



Implementing



End-Product

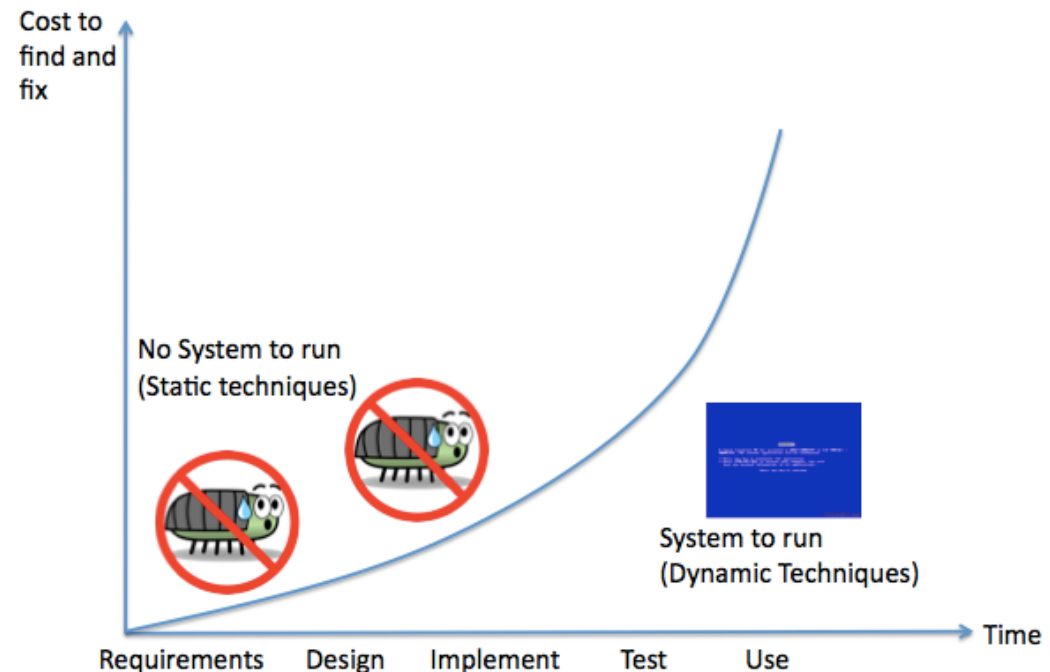


	Requirements	Design	Implementation	Product works as expected
Scenario 1	Correct requirements	Design meets requirements	Implementation meets design	Product works as expected
Scenario 2	Correct requirements	Design meets requirements	Mistakes made in building the system	Product has bugs
Scenario 3	Correct requirements	Mistakes made in design	Implementation to meet design	Product has flaws in design
Scenario 4	Mistakes in requirements	Design meets requirements	Implementation meets design	Product does not match users' requirements

Software Defect Reduction - some interesting facts

- Finding and fixing software problems after delivery is often 100 times more expensive than finding and fixing them during the requirements and design phase
- About 40-50 % of the effort on current software projects is spent on avoidable rework
- About 80 % of the avoidable rework comes from 20 % of the defects
- About 80 % of the defects come from 20 % of the models and about half of the modules are defect free
- About 90 % of the downtime comes from at most 10 % of the defects

From: Barry Boehm and Victor Basili,
Software Defect Reduction Top-10
List, in: Barry Boehm, Hans Dieter
Rombach, Marvin V. Zelkowitz,
Foundations of Empirical Software
Engineering: The Legacy of Victor R.
Basili



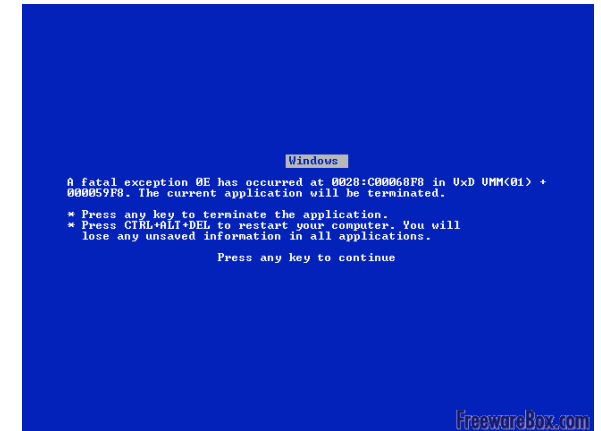
Important terminology



Humans make **mistakes/errors**, so do requirements engineers, designers, testers, and any other role in SE



This leads to **defects/bugs** in software artifacts (such as requirements, code, design, doc-ware)





When running the software certain actions by the user lead to a **failure**

Software quality (ISO/IEC

Characteristic	Description	Sub-Characteristics
Functionality	Delivery of functions satisfying implied/stated needs	Suitability, Accuracy, Security, Functionality Compliance, interoperability
Reliability	Capability of the software to maintain its level of performance under stated conditions/periods of time	Maturity, Fault tolerance, recoverability, reliability compliance
Usability	Effort needed to use and individual assessment of such use	Understandability, Learnability, Operability, Attractiveness, Compliance
Efficiency	Relationship between level of performance and the amount of resources used	Time behavior, resource utilization, compliance
Maintainability	Effort needed to modify the software based on specifications	Analyzability, Changeability, Stability, Testability, Compliance
Portability	Ability to transfer software between environments	Adaptability, Installability, Replaceability, Compliance

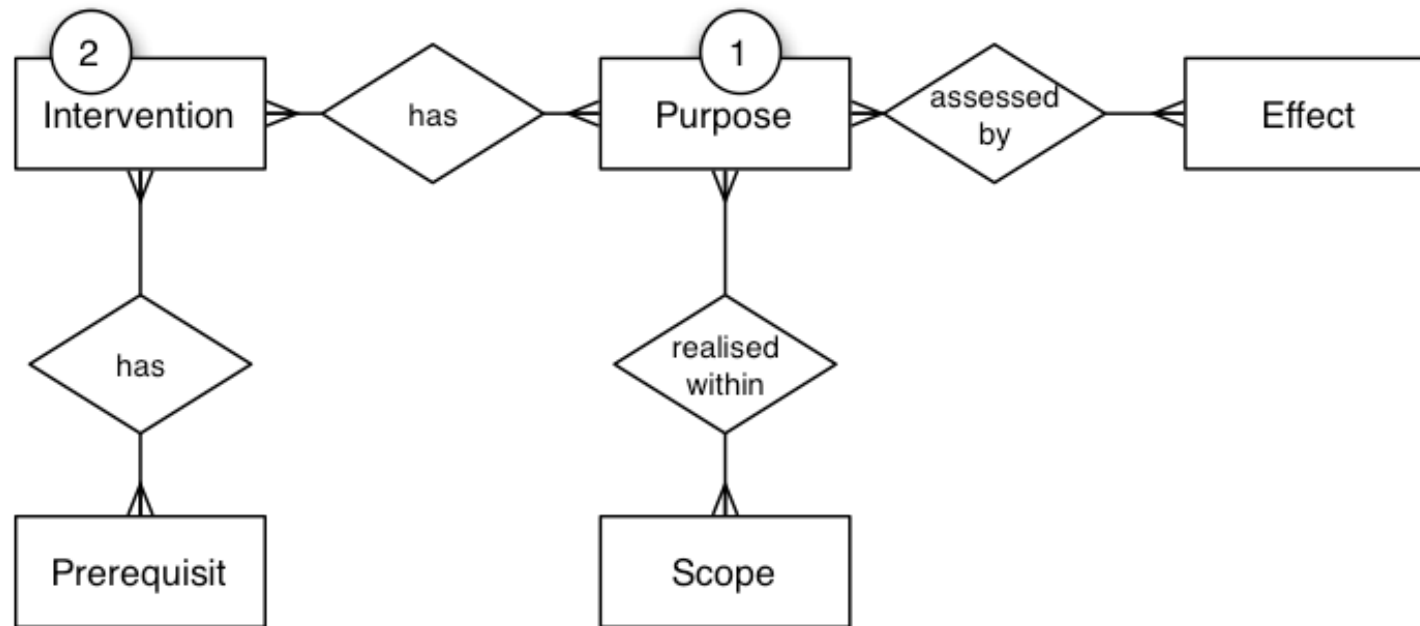
... some more viewpoints on quality

Characteristic	Software 	Tomatos 
Attributes of product	Characteristics of the product (see ISO)	Size, shape, color, taste, consistency
Fitness for use	Usefulness of the product (can you carry out your tasks with the features? Are there no useless features?)	Fits the receipt of the meal
Quality based on process	Follow “good” software development practices and have criteria (quality thresholds) for releasing	Good practices in growing and farming (e.g. no use of chemicals)
Value for Money	Stay in project budged (better to deliver less/what is really needed in comparison to delivering more of less quality)	Cheap, but good quality wrt. Fitness for use, attributes
Feelings/Image	This software is great! It’s beautiful and fun to use. I am “hot” and “up-to-date” when having this software (OS X? ;))	Great looks, taste, and service in delivery

Two main groups of techniques/methods to tack verification and validation

- **Static techniques:** Do not require an executable software artefact (e.g. inspections, reviews, walkthroughs)
- **Dynamic techniques:** Require an executable software artefact (software testing)

Structuring the area



Intervention = what you are doing to test/verify (can be a technique, process, etc.)

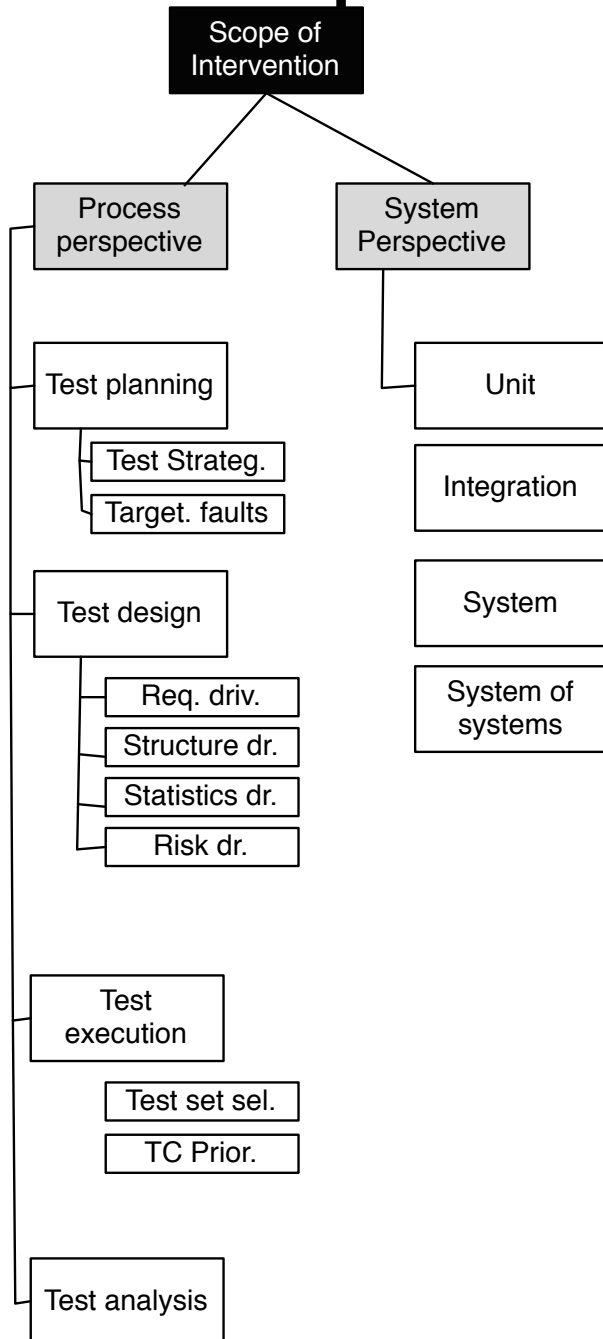
Purpose = why you are using the intervention

Effect = measurable variable to capture the effect of the intervention

Scope = Scope of the process and scope of the system

Prerequisite = what is needed to use the intervention

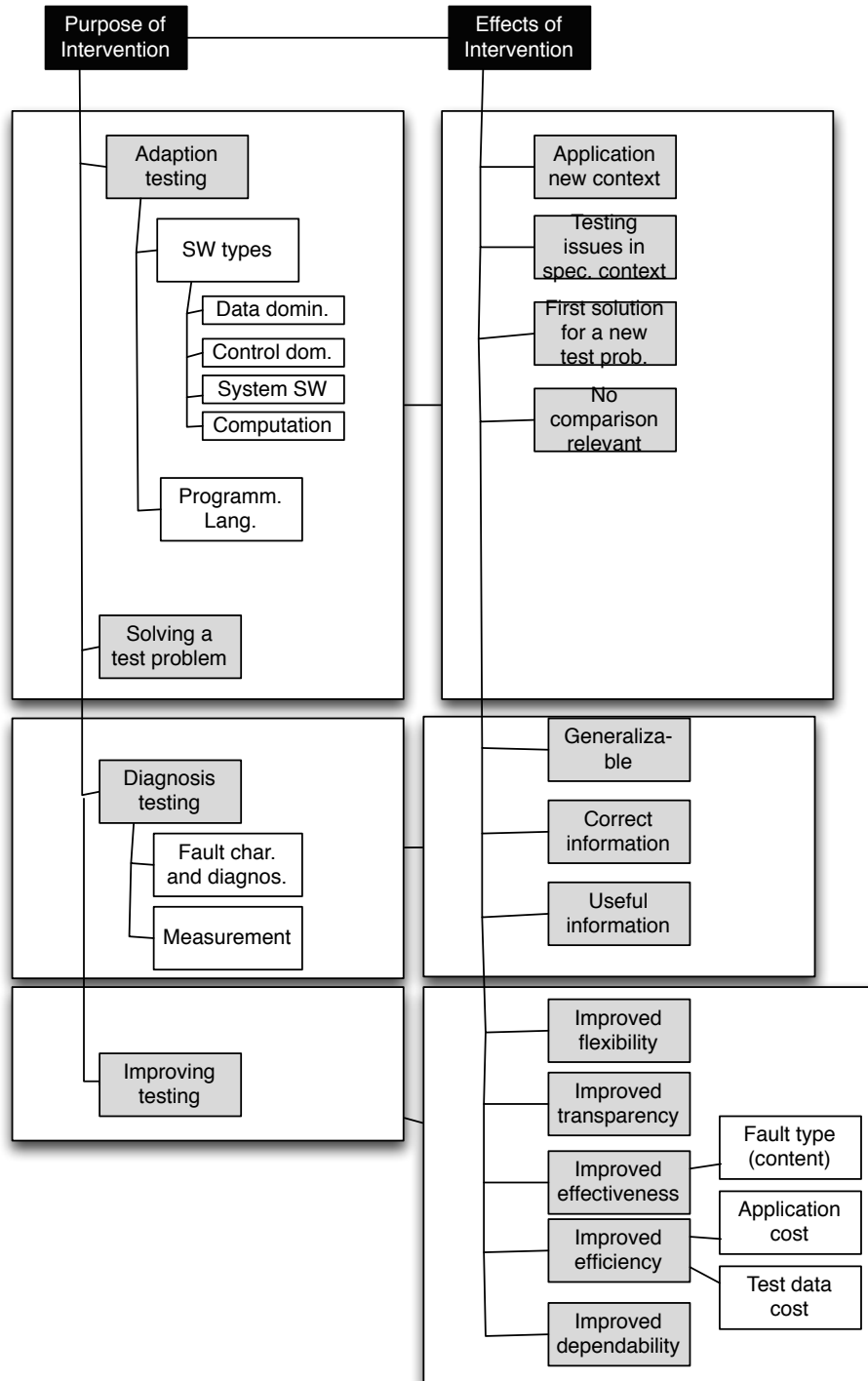
Scope of Intervention



Literature distinguishes three **main activities**: Test planning, test design, test execution, and test analysis

Tests are often focused on a specified scope, the **scope** so far is **not very well defined**, and we do not have a solution yet...

Purpose and effect



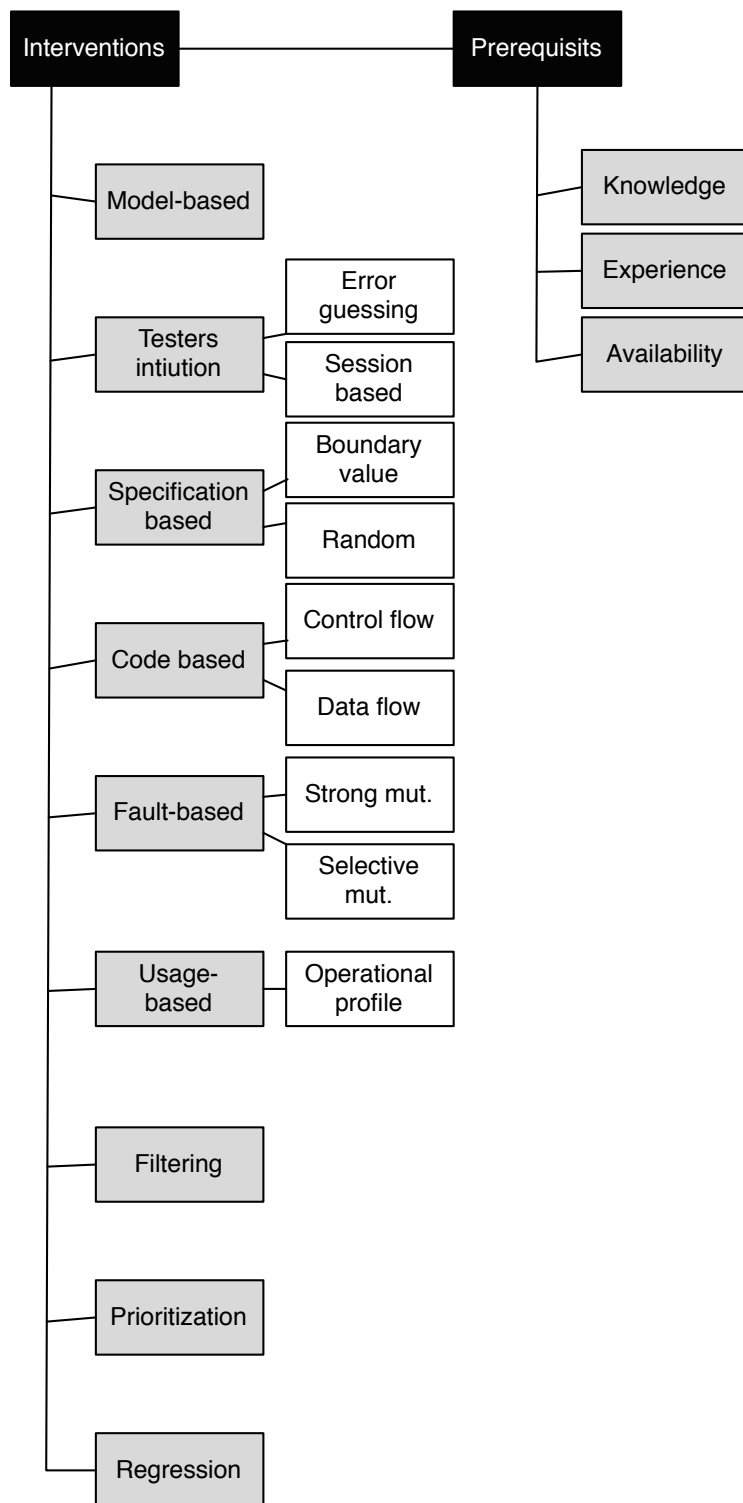
Purpose relates directly to effects, e.g. purpose is to adapt an existing testing intervention to test a new software type, i.e. ability to work in new context is of interest.

Multiple purposes can be defined for an intervention

Interventions and Prerequisites

Multiple interventions are proposed (e.g. specification based testing aiming at covering the specification uses different interventions, such as boundary value analysis)

Prerequisites might be given to apply a specific technique, e.g. knowledge and experience might play a major role in intuition based testing, or specific infrastructure needs to be available



About the course - learning outcomes

• Knowledge and Understanding (KU)

- KU1: Describe and classify techniques within verification and validation
- KU2: Ability to describe the techniques
- KU3: Classify techniques and being able to compare them
- KU4: Ability to select relevant works and information within verification and validation to solve practical research problems

• Skills and Abilities (SA)

- SA1: In a team to create and implement a test plan.
- SA2: Individually apply the introduced techniques for static verification, and testing techniques to real software systems.

• Critical reflection (CR)

- CR1: Critically reflect and discuss a controversial topic in verification and validation.
- CR2: Ability to critically evaluate the strengths and weaknesses of verification and validation techniques

Course moments and their relation to the objectives

Moments	Relation to outcomes	Description	Mandatory
Lectures	KU1 to KU4	In the lectures an overview of the structure and the techniques is given	No, but strongly recommended to pass the course
Exercises	KU1 to KU4, CR1 and CR2	Each lecture comes with a set of questions that you should answer	No, the exercises are, however, a great way to prepare for the examination
Project	KU3, KU4, SA1, SA2, CR1, CR2	You are to conduct an automated inspection of source code of a system and write a test plan	Yes, one submission and two re-submissions
Exam	KU1 to KU4, CR1 and CR2	The questions in the exercises will be similar to those in the exam	Yes, per year three examinations are offered.
Debate	KU1 to KU4, CR1, CR2	A debate shall be held where two teams discuss a controversial topic, a debate report is to be prepared	Yes, attendance and reflection report are mandatory

Moments with attendance

	Topic	Date
L01	Introduction	2014-01-21
L02	Intro + Dynamic V&V and Static V&V	2014-01-28
L03	Coverage and Intervention 1: Specification-based	2014-02-04
L04	Coverage and Intervention 2: Code-based	2014-02-11
Debate	Test last development vs. Test first development	2014-02-13
L05	Coverage and Intervention 3: Fault-based and Usage-based	2014-02-18
L06	Test case generation	2014-02-25
L07	Test case execution	2014-03-04
L08	Test analysis	2014-03-11

*New students will arrive next week, hence part of the lecture will be repeated for these students, hence only the second hour will be of relevance for you.

Main take-aways

- Difference between verification and validation
- Motivation for conducting verification and validation activities
- Terminology
- Overall structure of the area (when looking at techniques, think about where they belong to in terms of scope, possible effects, etc.)
- Please solve “Exercises 1” to familiarize with the content of the lecture