Test Design Techniques on One Page

Black-box (models, interfaces, data)

Standards (e.g. ISO/IEC 9126/25000, IEC 61508), norms, (formal) specifications, claims Requirements-based with traceability matrix (requirements x test cases) **Use case-based te**sting (sequence diagrams, activity diagrams) CRUD (Create, Read, Update, Delete) (data cycles, database operations) Flow testing, scenario testing, soap opera testing User / Operational profiles: frequency and priority / criticality (Software Reliability Engineering) Statistical testing (markov chains) Random (monkey testing) Features, functions, epics, user stories, processes, services, interfaces Design by contract (built-in self test) Equivalence class partitioning Domain partitioning, category-partition method Classification-tree method Boundary value analysis Special values Test catalog / matrix for input values, input fields State-based testing (Finite State Machines) Cause-effect graphing Decision tables, decision trees Syntax testing (grammar-based testing) Combinatorial testing (orthogonal / covering arrays, pair-wise, n-wise*)* Time cycles (frequency, recurring events, test dates) Evolutionary testing Metamorphic testing

Grey-box

**Dependencies / Relations between classes**, objects, methods, functions Dependencies / Relations between components, services, applications, systems Communication behavior (dependency analysis) **Trace-based testing passive te**sting) Protocol based (sequence diagrams, message sequence charts)

ܟܠܗܗܗ ܟܢܗܗܗܗܠܩ ܗܘܘ ܗܗܗܓܣܚ ܣܩ ܗܗܗ ܩܢܩܗܩܗ

Control flow-based

White-box

(internal structure, paths)

Coverage (specification-based, model-based, code-based)

NOT

Statements (CO), nodes Branches (C1), transitions, links, paths Conditions, decisions (C2, C3) Elementary comparison (MC/DC) Interfaces (S1, S2) Cyclomatic complexity (McCabe) Metrics (e.g. Halstead) Read / Write access **IDef! Use criteria**

Static metrics

Data flow-based

Positive, valid cases Negative, invalid cases

Normal, expected behavior Invalid, unexpected behavior Error handling Exceptions

Fault-based

Risk-based Systematic failure analysis (Failure Mode and Effect Analysis, Fault Tree Analysis) Attack pattems (e.g. by James A. Whittaker, Jon Hagar) Error catalogs, bug taxonomies (e.g. by Boris Beizer, Cem Kaner) Bug patterns: standard, well-known bug patterns or produced by a root cause analysis Bug reports Fault model dependent on used technology and nature of system under test Test patterns (e.g. by Robert Binder), Questioning patterns (Q-patterns by Vipul Kocher) Ad hoc, intuitive, based on experience, check lists Error guessing Exploratory testing, heuristics, mnemonics (e.g. by James Bach, Michael Bolton) Fault injection Fuzzing Mutation testing

Regression (selective retesting) Retest all

Retest by risk, priority, severity, criticality Retest by profile, frequency of us**age, parts which are often used** Retest changed parts Retest parts that are influenced by the changes (impact analysis, dependency analysis)

**Key Categorization** Methods, Paradigms, Techniqu**es, Styles, and Ideas to Create a Test Case** Effort / Difficulty / Resulting Test Intensity (5 Levels)

Peter Zimmerer ©Siemens AG, Corporate Technology

**SIEMENS**