Test Design Techniques on One Page

Black-box (models, interfaces, data)	Standards (e.g. ISO/IEC 9126/25000, IEC 61508), norms, (formal) specifications, claims	3
	Requirements-based with traceability matrix (requirements x test cases)	3
	Use case-based testing (sequence diagrams, activity diagrams)	3
	CRUD (Create, Read, Update, Delete) (data cycles, database operations)	3
	Flow testing, scenario testing, soap opera testing	4
	User / Operational profiles: frequency and priority / criticality (**S**oftware **R**eliability **E**ngineering)	4
	Statistical testing (markov chains)	4
	Random (monkey testing)	4
	Features, functions, epics, user stories, processes, services, interfaces	1
	Design by contract (built-in self test)	3
	Equivalence class partitioning	2
	Domain partitioning, category-partition method	4
	Classification-tree method	3
	Boundary value analysis	2
	Special values	1
	Test catalog / matrix for input values, input fields	5
	State-based testing (**F**inite **S**tate **M**achines)	3
	Cause-effect graphing	5
	Decision tables, decision trees	5
	Syntax testing (grammar-based testing)	4
	Combinatorial testing (orthogonal / covering arrays, pair-wise, n-wise)	3
	Time cycles (frequency, recurring events, test dates)	4
	Evolutionary testing	5
	Metamorphic testing	3

Grey-box	Dependencies / Relations between classes, objects, methods, functions	2
	Dependencies / Relations between components, services, applications, systems	3
	Communication behavior (dependency analysis)	3
	Trace-based testing (passive testing)	3
	Protocol based (sequence diagrams, message sequence charts)	4

White-box (internal structure, paths)	Control flow-based	Coverage (specification-based, model-based, code-based)	Statements (C0), nodes	2
			Branches (C1), transitions, links, paths	3
			Conditions, decisions (C2, C3)	4
			Elementary comparison (MC/DC)	5
			Interfaces (S1, S2)	4
		Static metrics	Cyclomatic complexity (McCabe)	4
			Metrics (e.g. Halstead)	4
	Data flow-based		Read / Write access	3
			Def / Use criteria	5

Positive, valid cases	Normal, expected behavior	1
Negative, invalid cases	Invalid, unexpected behavior	3
	Error handling	3
	Exceptions	5

Fault-based	Risk-based	2
	Systematic failure analysis (**F**ailure **M**ode and **E**ffect **A**nalysis, **F**ault **T**ree **A**nalysis)	4
	Attack patterns (e.g. by James A. Whittaker, Jon Hagar)	3
	Error catalogs, bug taxonomies (e.g. by Boris Beizer, Cem Kaner)	4
	Bug patterns: standard, well-known bug patterns or produced by a root cause analysis	3
	Bug reports	2
	Fault model dependent on used technology and nature of system under test	2
	Test patterns (e.g. by Robert Binder), Questioning patterns (Q-patterns by Vipul Kocher)	3
	Ad hoc, intuitive, based on experience, check lists	1
	Error guessing	2
	Exploratory testing, heuristics, mnemonics (e.g. by James Bach, Michael Bolton)	2
	Fault injection	4
	Fuzzing	3
	Mutation testing	5

Regression (selective retesting)	Retest all	5
	Retest by risk, priority, severity, criticality	2
	Retest by profile, frequency of usage, parts which are often used	3
	Retest changed parts	2
	Retest parts that are influenced by the changes (impact analysis, dependency analysis)	5

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

Peter Zimmerer
©Siemens AG, Corporate Technology
SIEMENS