Use of Domain-Specific Modeling with Model-Based Testing

Juha-Pekka Tolvanen, PhD, MetaCase

Stephan Schulz, PhD, Conformiq

Contents

- Introduction to DSLs and MBT
- DSL + MBT = ?
- Case 1: Web application (IT)
- Case 2: Military radio (embedded)
- Results
- How to get started
- Summary, Q&A

Some relevant language classifications to start with

- General-Purpose / **Domain-Specific**
  - Narrow area of interest
  - Can be inside one company and its products only

Narrow area of interest

- Example: Calendar application

Generic  Specific

```
@Test
  public void addTask() {
    CalendarUser user = new CalendarUser();
    CalendarApplication calendar = user.getCalendar();
    Calendar time = Calendar.getInstance();
    time.set(2012, Calendar.FEBRUARY, 2);
    CalendarTask calendarTask =
      calendar.addTask(time.getTime(),"My Little Task");
    assertEquals("Number of tasks", 1,
                 calendar.getTasks().size());
    assertEquals("Task description”, "My Little Task",
                 calendarTask.getDescription());
    assertEquals("Task time”, time.getTime(),
                 calendarTask.getWhen());
  }
```

Some relevant language classifications to start with

- General-Purpose / **Domain-Specific**
  - Narrow area of interest
  - Can be inside one company and its products only
- **Problem Domain** / Solution Domain
  - Higher abstraction as it leads to improved productivity

Problem domain

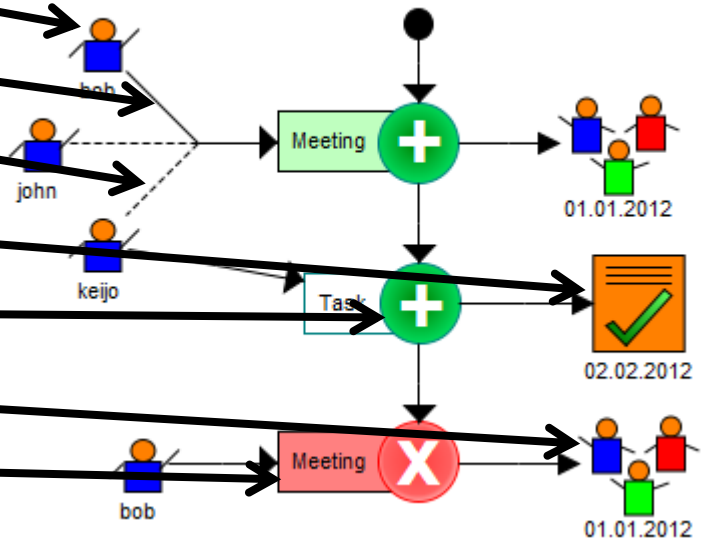
- Language concepts = domain concepts
- In calendar domain:
  - Meeting
  - Task
  - Person
  - Organizer
  - Participant
  - etc.
- Raise the level of abstraction

Some relevant language classifications to start with

- General-Purpose / **Domain-Specific**
  - Narrow area of interest
  - Can be inside one company and its products only
- **Problem Domain** / Solution Domain
  - Higher abstraction as it leads to improved productivity
- **Graphical** / Text / Matrix / Table etc.
  - Always apply style close to the domain's natural representation
  - In this talk we apply graphical modeling languages
    - Humans are good at spotting visual patterns
    - Easier to read, understand and communicate with
    - Expressing conditions, parallelism and structures
    - Reusability

Graphical modeling languages

- Language concepts = domain concepts:
  - Person
  - Organizer
  - Participant
  - Task
    - Add, remove,…
  - Meeting
    - Add, remove,…
  - etc.



- Domain rules in the language
  - Only organizer can cancel the meeting, etc.

Some relevant language classifications to start with

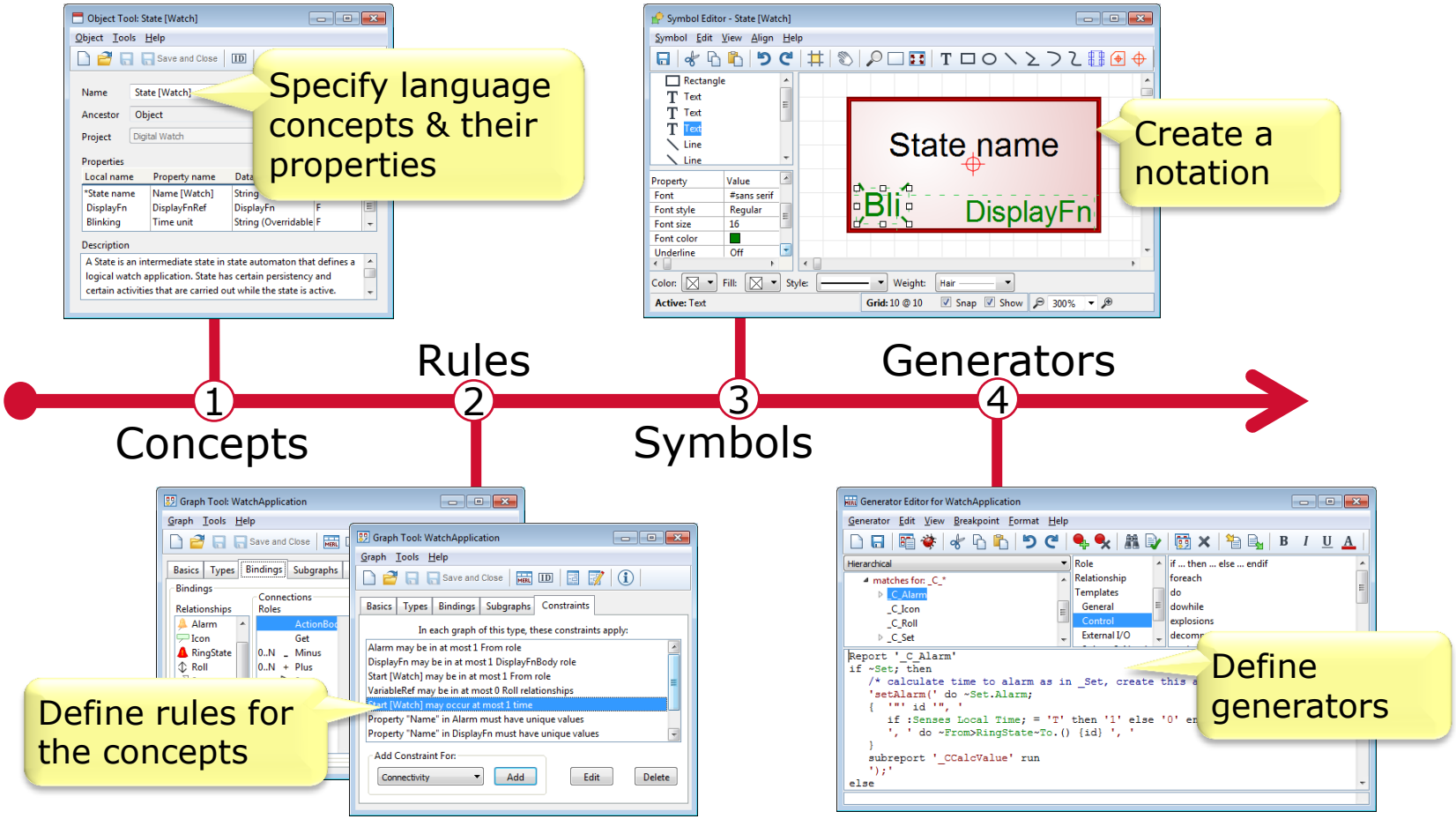
- General-Purpose / **Domain-Specific**
  - Narrow area of interest
  - Can be inside one company and its products only
- **Problem Domain** / Solution Domain
  - Higher abstraction as it leads to improved productivity
- **Graphical** / Text / Matrix / Table etc.
  - Always apply style close to the domain's natural representation
- **Static structures** / **Behavior**

Domain-Specific Modeling Languages

- Applied in particular for automating repetitive development efforts:
  - Product line development
  - Platform-based application development
  - Product configuration and deployment
- Higher abstraction and automation (code generation) leads to significant results:
  - 5-10x improvements in productivity*
  - Better quality as errors can be detected or avoided already in the design phase*

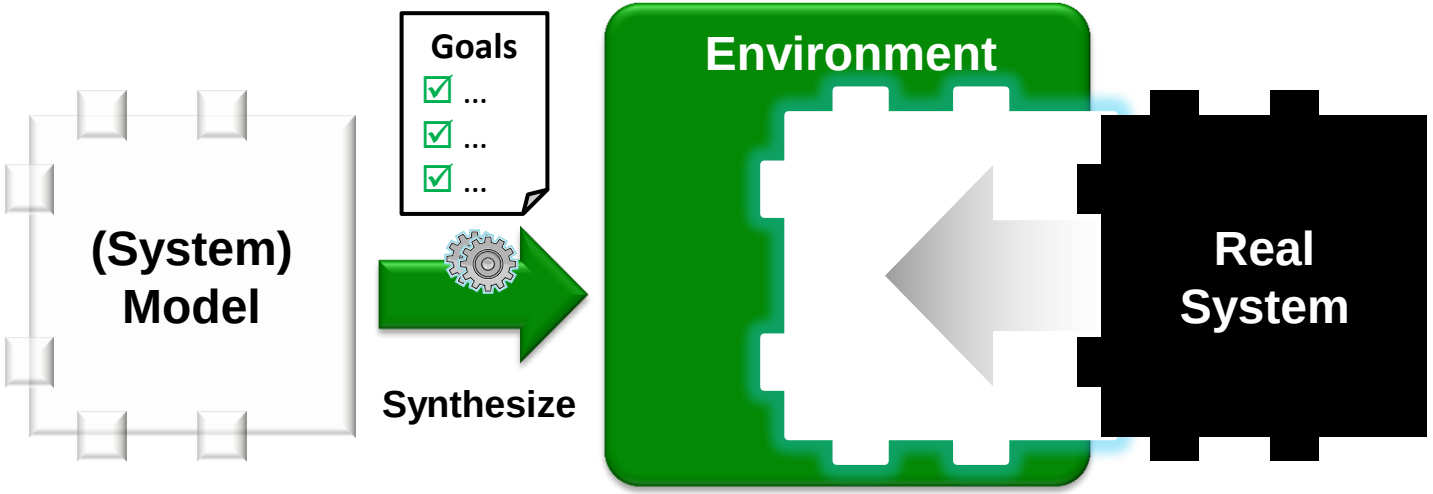
* See references on EADS, NSN, Nokia, Panasonic, Polar Elektro, USAF

Steps for Defining Domain-Specifc Modeling Languages and Generators



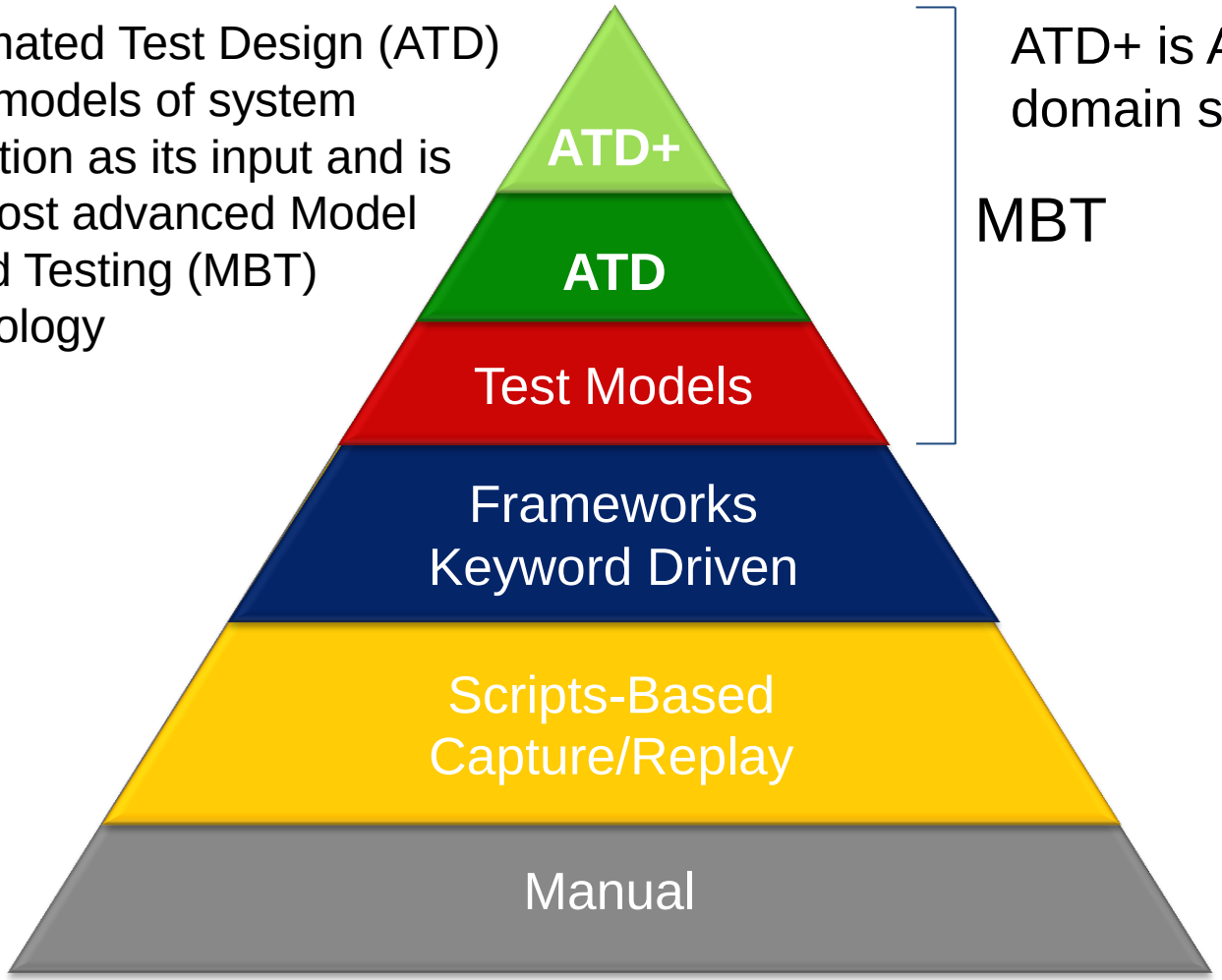
About Model-Based Testing (MBT)

- Umbrella term for using models in a testing context
- One approach is to use MBT for automating *test design*
  - Here model reflects operation of the system to be tested
  - MBT *complements* test execution
  - Recognized by worldwide industrial standards (ETSI)



Evolution of Software Testing

Automated Test Design (ATD) uses models of system operation as its input and is the most advanced Model Based Testing (MBT) technology



ATD+ is ATD driven by a domain specific language

MBT

Test Approach Comparison Heat Map

Test Approach	Test Coverage	Early Problem Discovery	Functional Complexity	Test Artifact Reuse	Required Skill Set	Test Process Optimization	Productivity Gain Initial	Productivity Gain Iteration
Manual Test	2	2	2	0	2	1	1	1
Test Scripts	5	5	6	6	7	4	4	3
Test Modeling	7	5	5	4	5	6	7	6
Automated Test Design	10	8	8	8	8	8	6	8
DSL Driven ATD	10	8	8	9	4	8	8	9

ATD+: DSL driven MBT

- Draws from all benefits of conventional ATD
  - Automated test design and traceability
  - Integration into test automation ecosystem
  - 5x improvements in productivity
- Enables testers to model system operation
  - No longer programming skills required
  - Less training and faster ramp up
- Allows other stakeholders to review models
  - “Shift (really) left” … engage your customer!

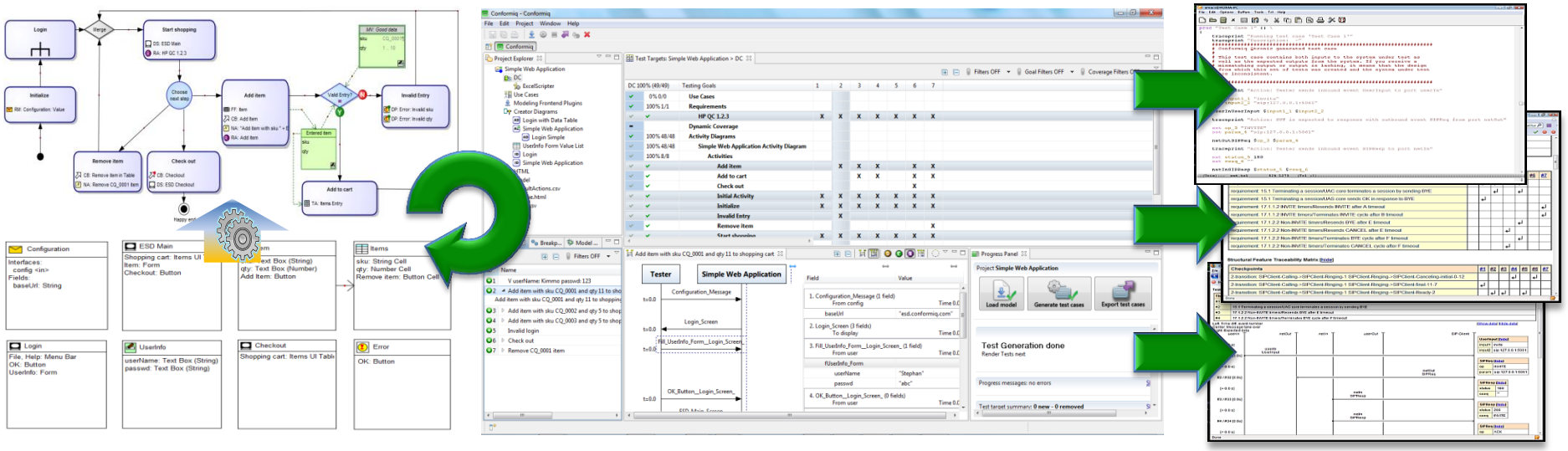
~5x (DSL) combined with ~5x (ATD) = **???**

Automated Test Design Workflow

Model
System Operation

Direct & Review
Test Design

Generate Test Scripts
& Documentation

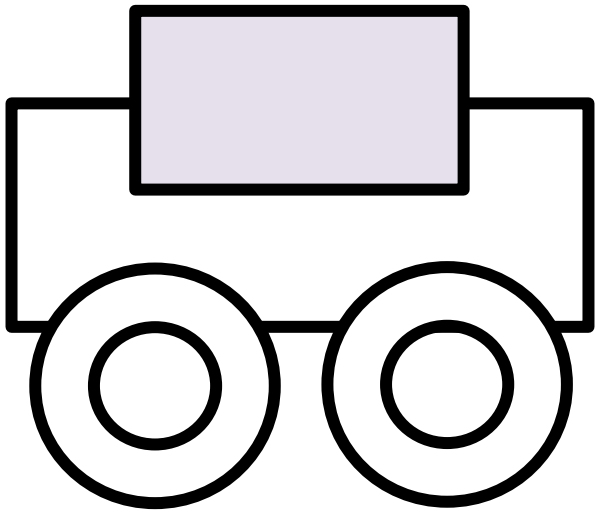


Domain Specific
Modeling Tool

Model Based
Test Design Tool

Test Execution
Tool(s)

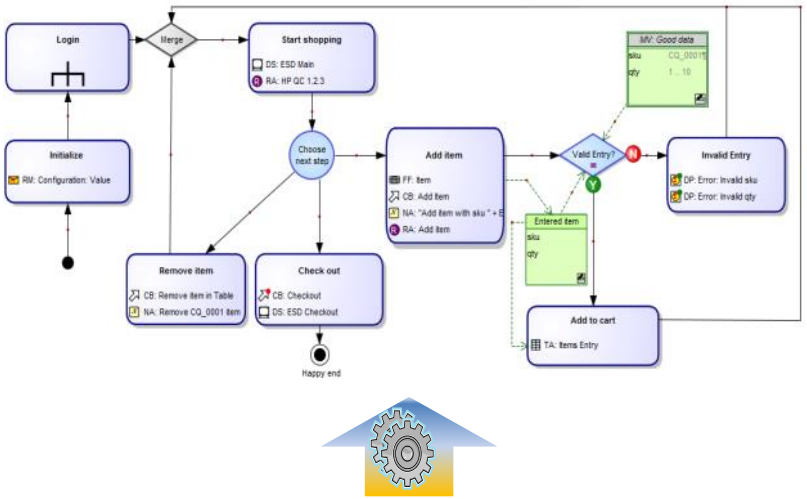
Why are DSLs so Important in Testing?

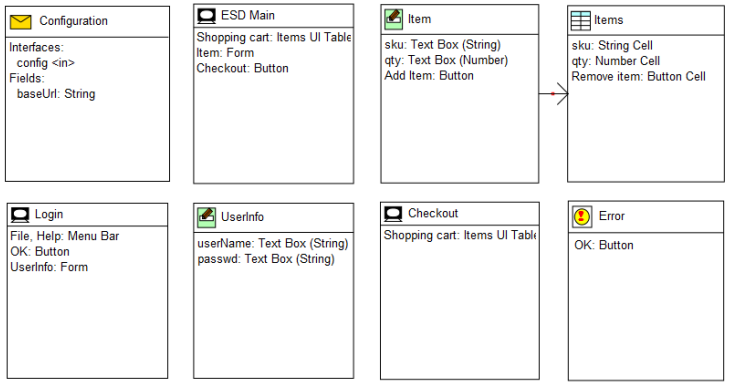


Testing is about achieving a common understanding

Case 1: Conformiq Creator

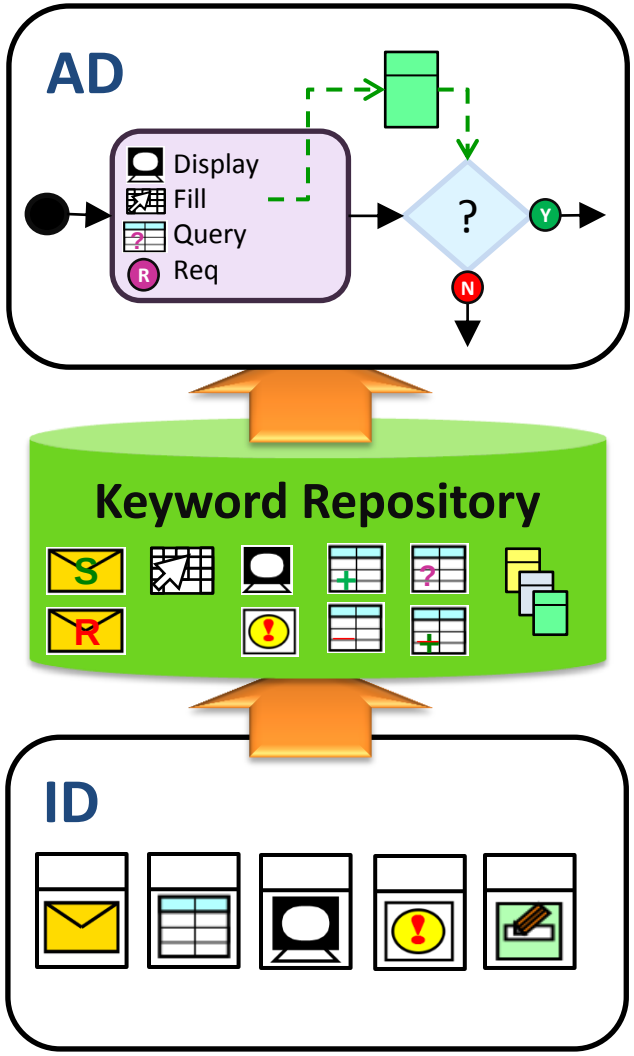
- A DSL developed for
  - Modeling system operation in later testing phases such as *system & end-to-end testing*
  - BFSI, Enterprise IT, web services, web applications, etc.
  - Testers and Subject Matter Experts
- Encodes best practice
  - Provides set of pre-defined modeling building blocks



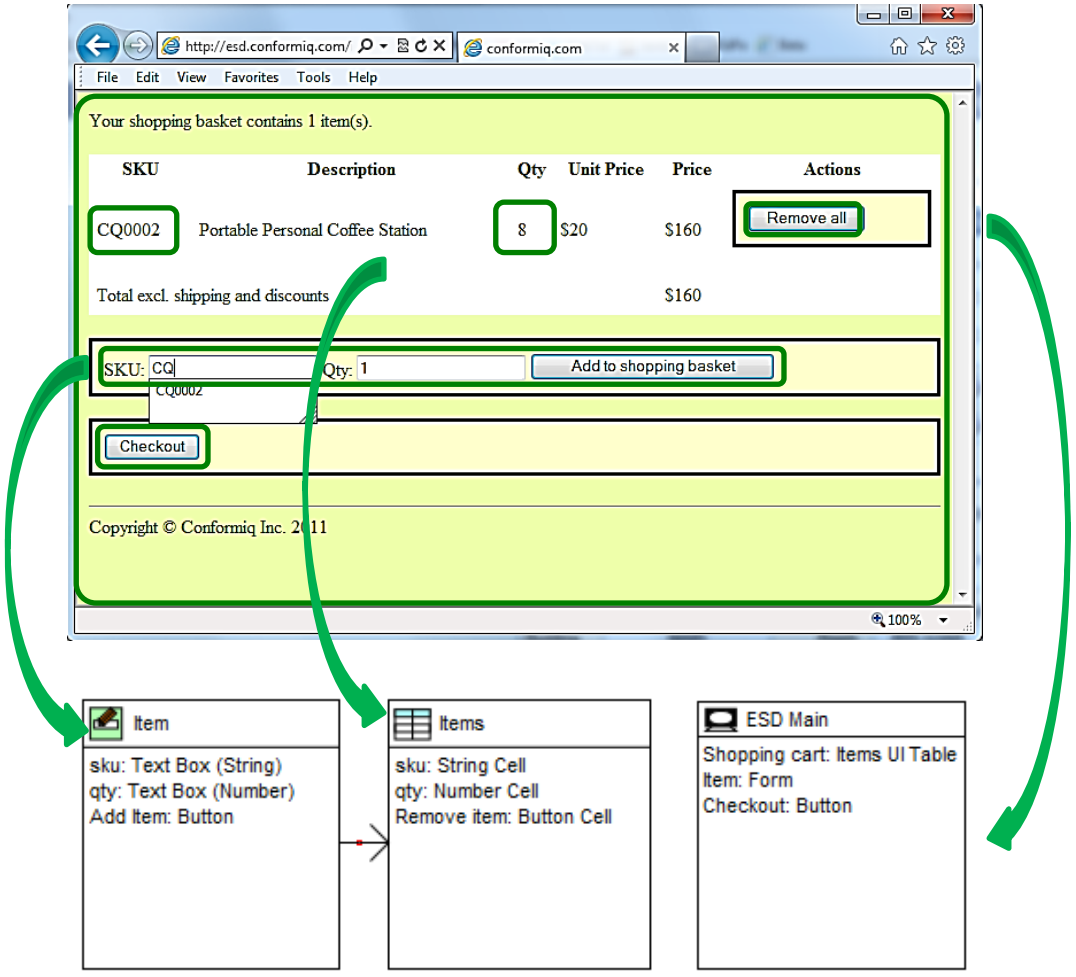


Modeling with Creator

- Activity Diagrams
  - Specify system operation using standard activity diagram symbols
  - Refine activities and decision based on action keywords and data objects
- Keyword Repository
  - Action keywords and data objects generated from interface objects
- Interface Diagrams
  - Define external SUT interfaces based on domain specific pre-defined interface objects



About Interface Diagrams

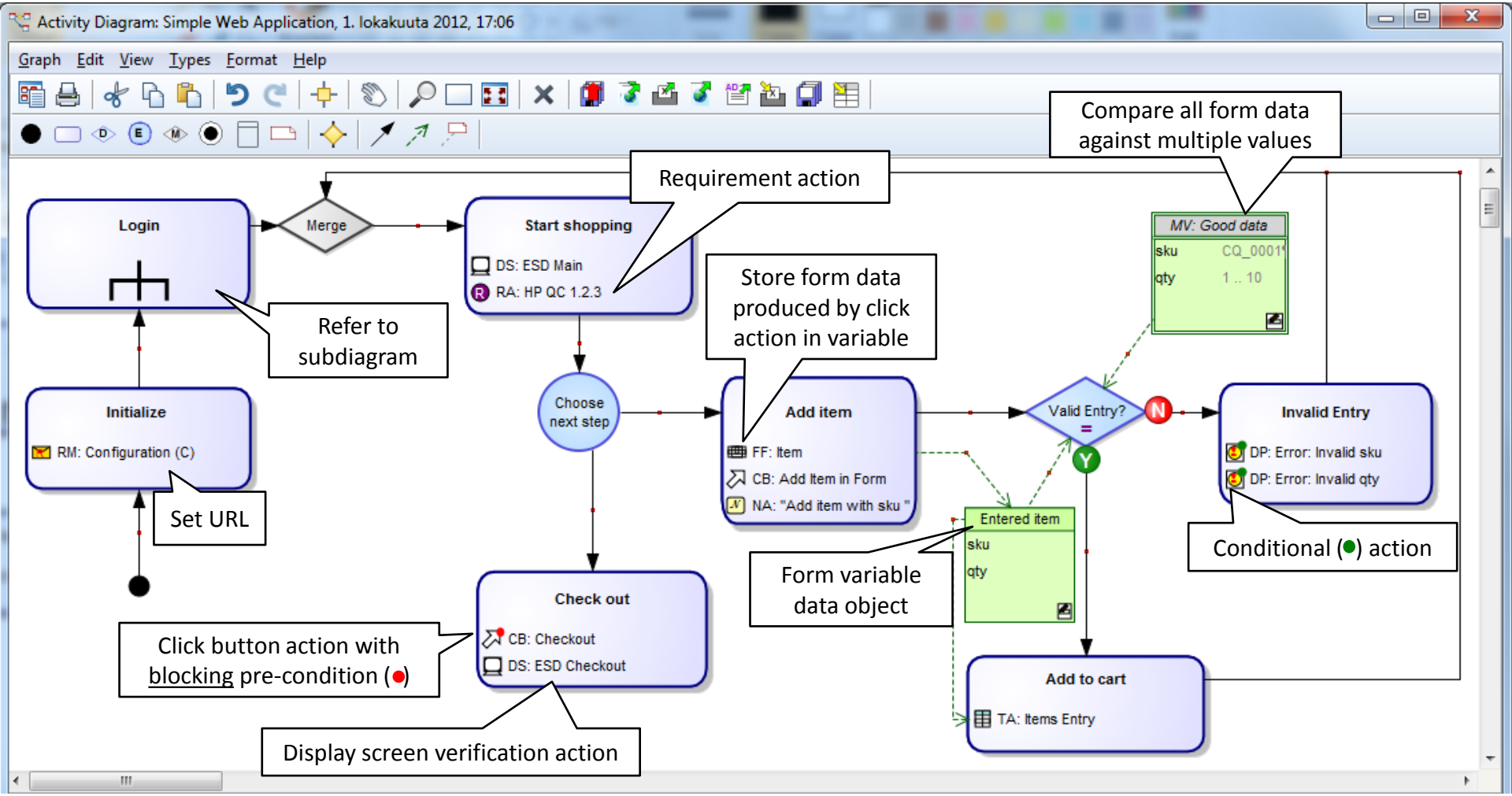


About Activity Diagrams

Fulfill a dual purpose:

- Specifies “what” is to be tested, i.e., relevant system operation in terms of workflows
  - Using activity, decision, event, merge nodes and control flow
- Specifies “how” to test based on action keywords and data objects generated from interface diagrams
  - Actions refine the activity description
  - (Graphical) conditions refine decisions
  - Data flows

Activity Diagram Example

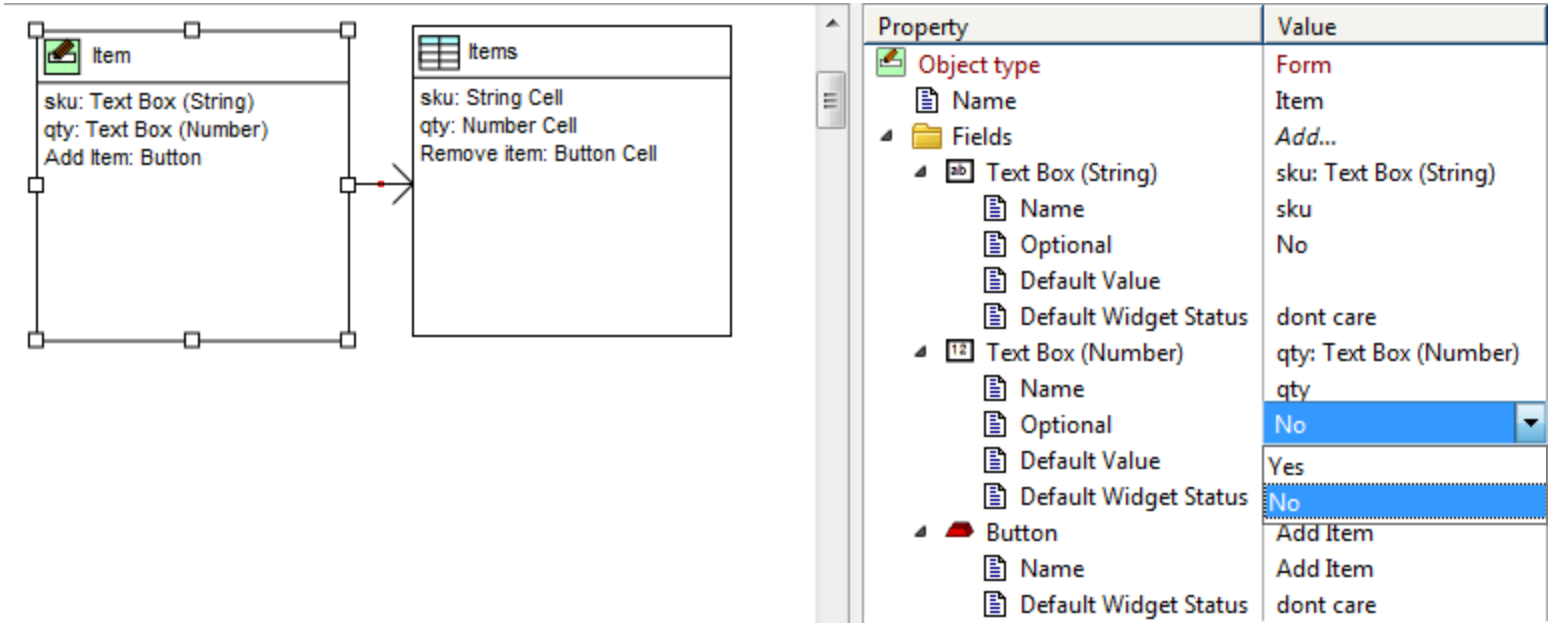


Generic vs Domain Specific

Generic Concept	Domain Specific Concept
Class	Message, Screen, Button
integer, boolean, String	Number, Checkbox, Dropdown Box
Receive on a port	Click a button, fill a form, Receive a message
Send from a port	Display a screen, Send a message
Compare each field of a variable to basic value	Compare entire message or form variable against value

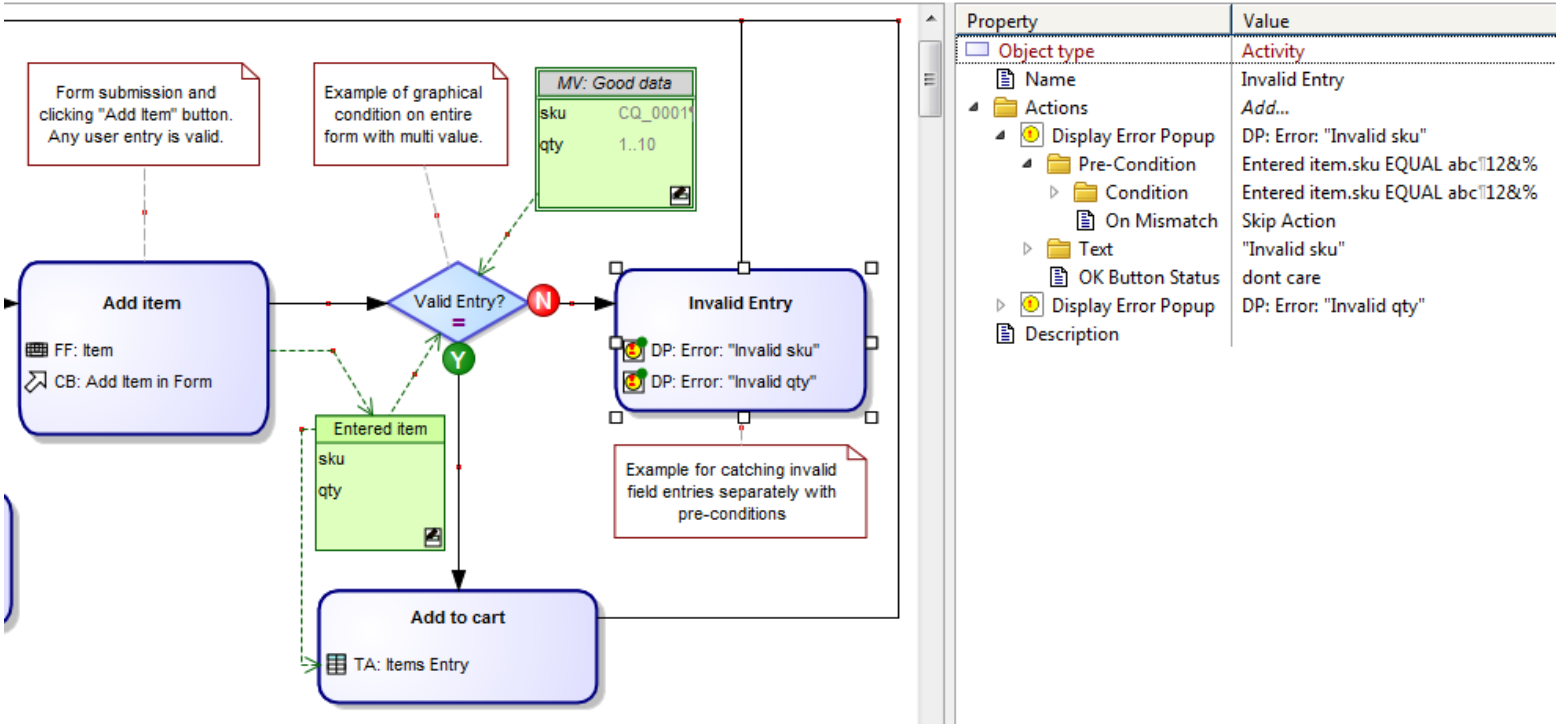
Note: Domain = Application Domain and Testing Domain!

Idea: Simplify, Reduce & Reuse



- Symbols have look & feel closer to application domain
- Abstraction and layering of model information
  - Not all model information is on the canvas
- Object driven specification enables reuse
- Less modeling errors by using “specification by selection”

Modeling for Testing



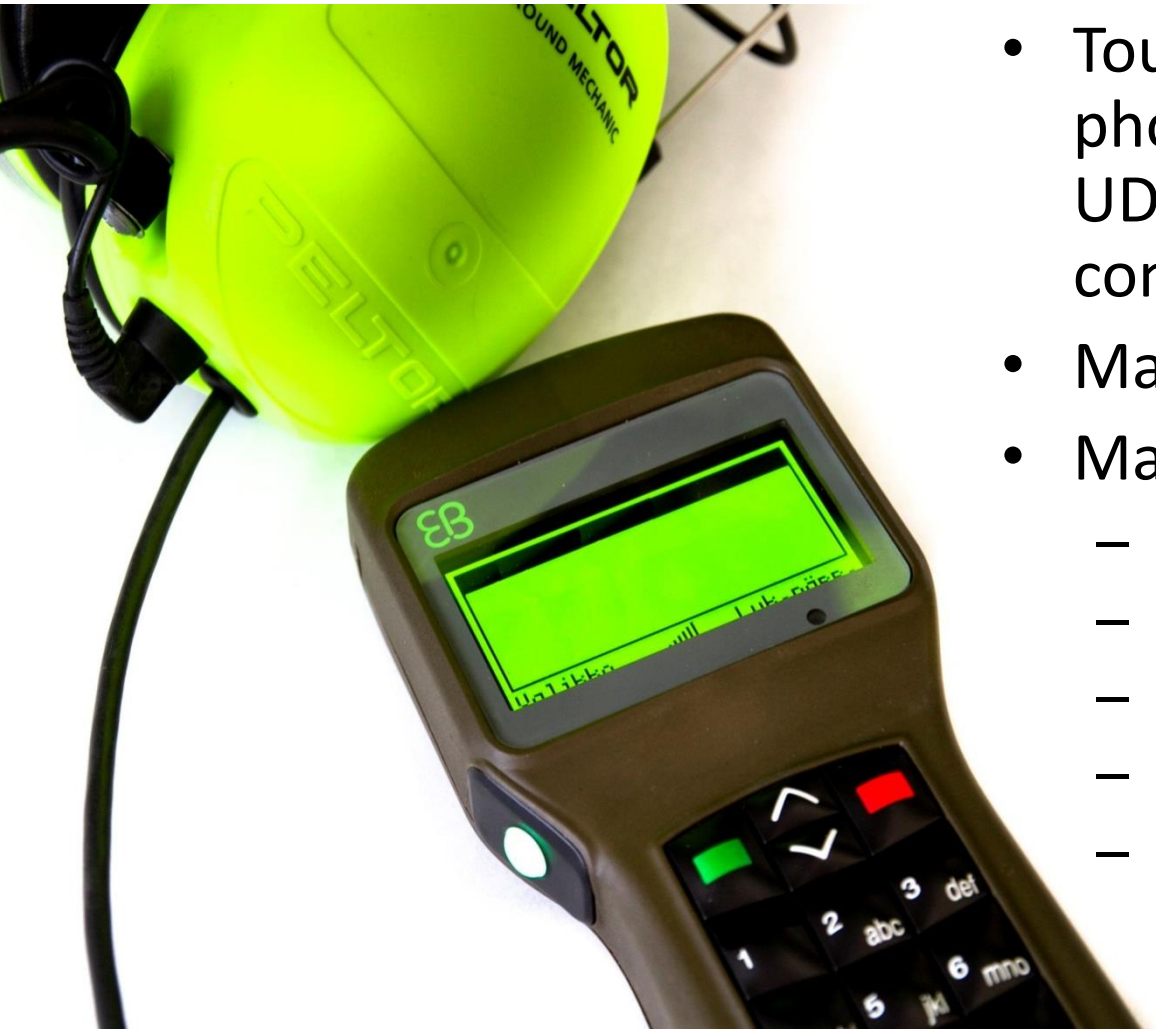
- Work with complete data object values
- Enable use wildcards
- Visual indication of pre-conditions

1st Industrial Feedback on Creator

- Doubled productivity over conventional UML/ Java based automated test design solution
- Training need reduced from 4 weeks to 4 days
- Subject Matter Experts (SMEs) and manual testers are able to model for testing
- Ecosystem from conventional automated test design approach could be reused

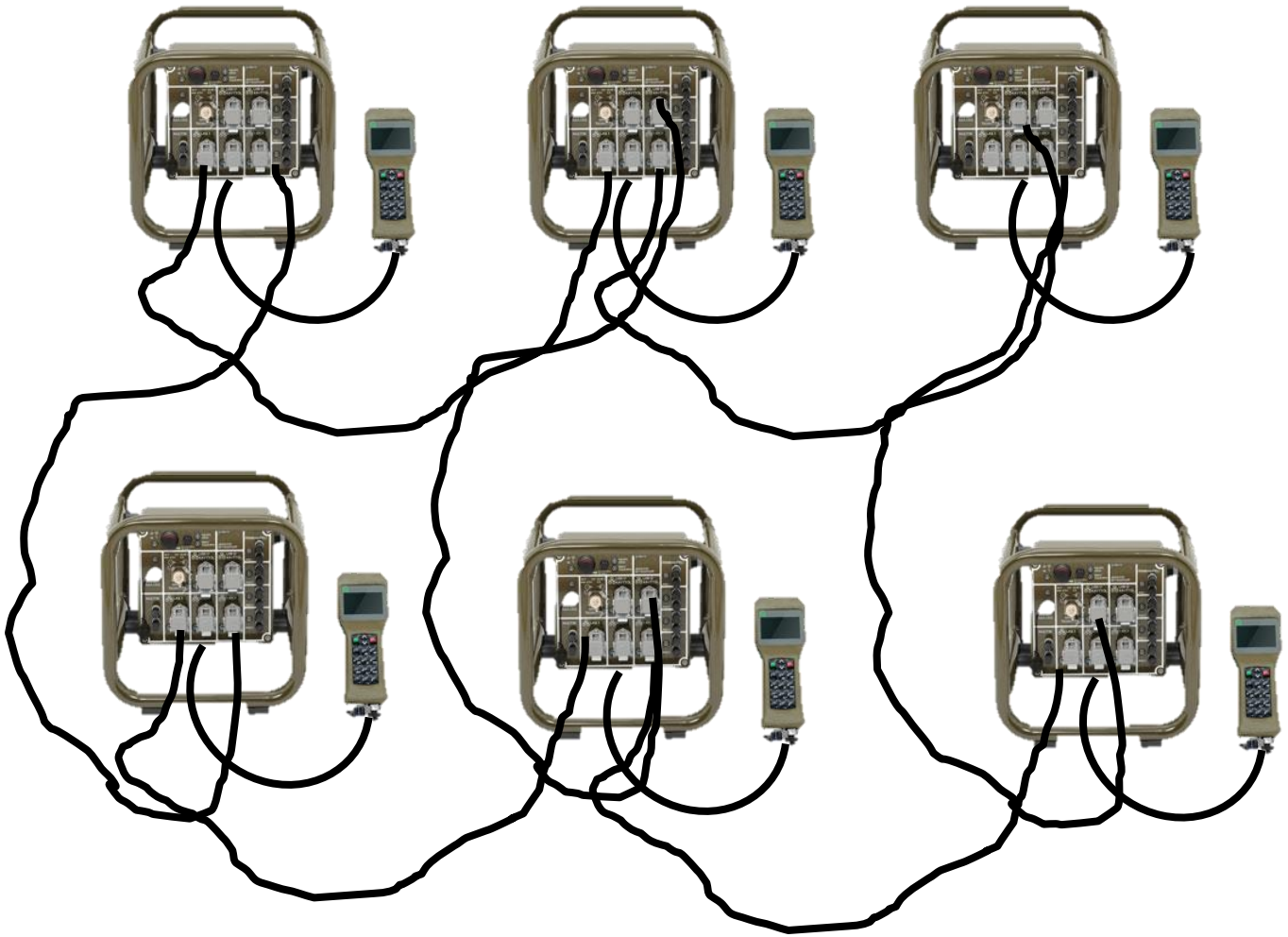
Case 2: Elektrobit Military radio (Puolitaival et al., 2011)

EB Tough VoIP Features



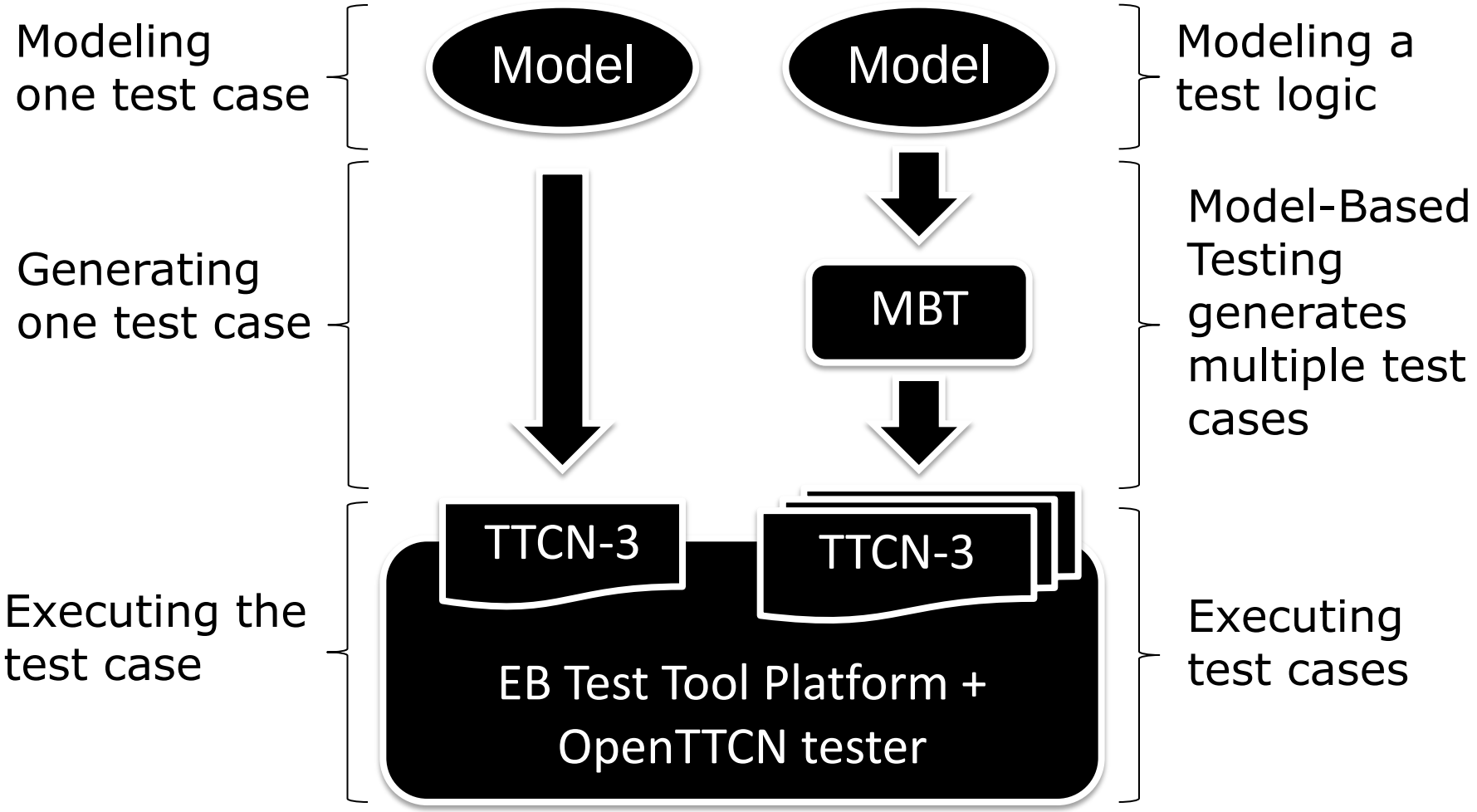
- Tough VoIP is a wired phone that is using UDP/IP network for connection
- Manufacturer: Elektrobit
- Main features:
  - Easy configuration
  - Point-to-Point call
  - All call
  - War-proof device
  - As simple as possible

Testing problem

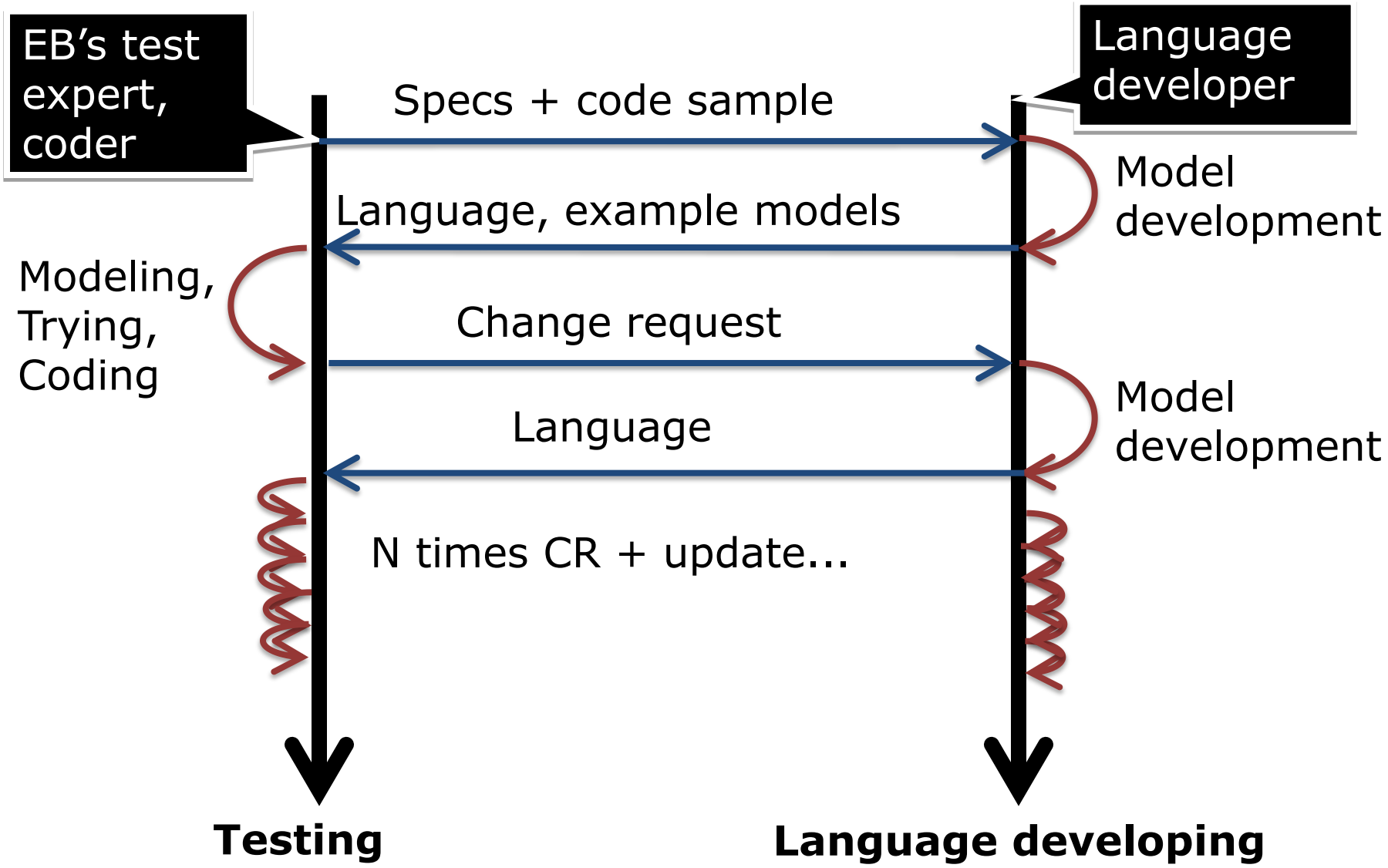


ETC...

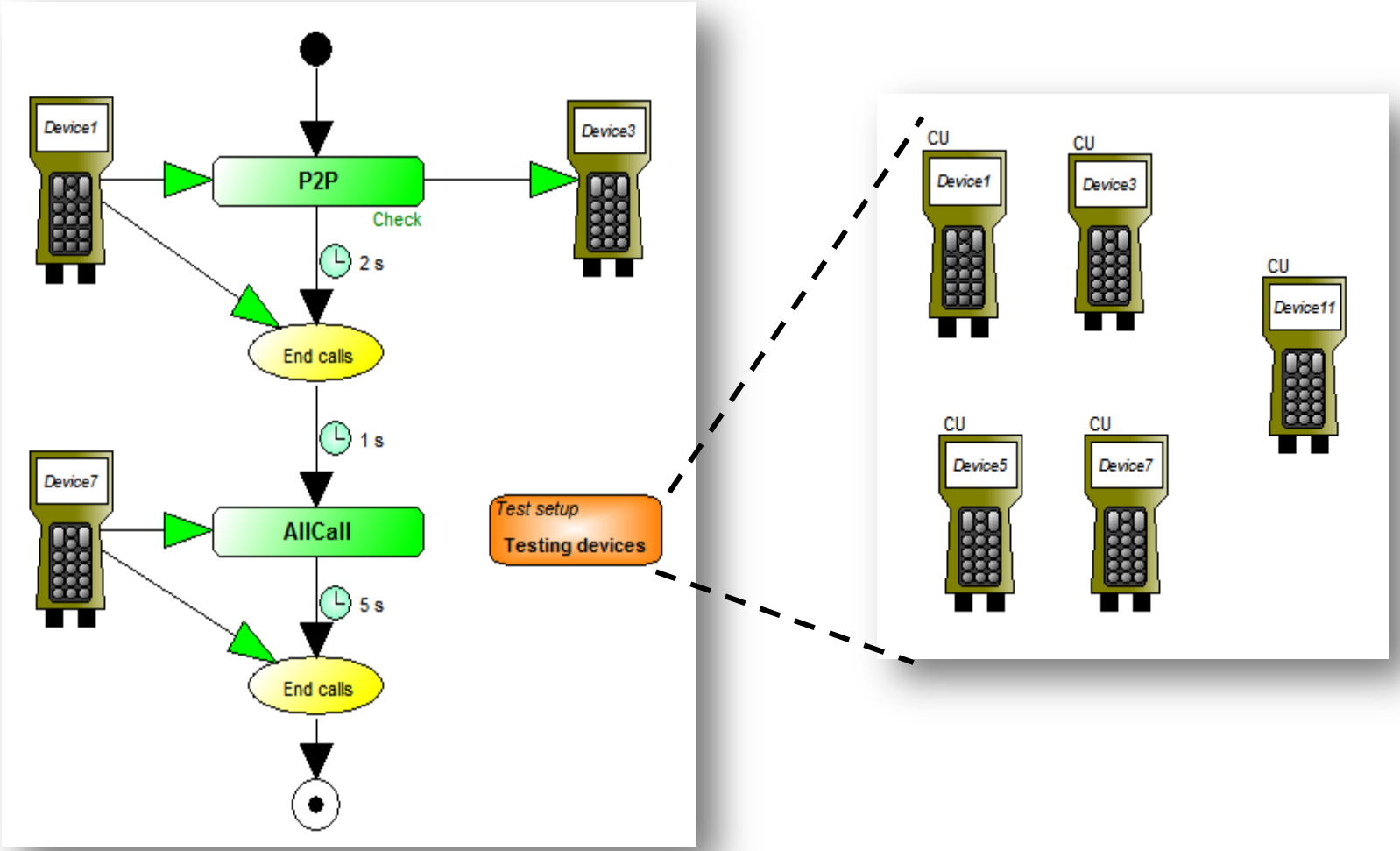
Two language solution



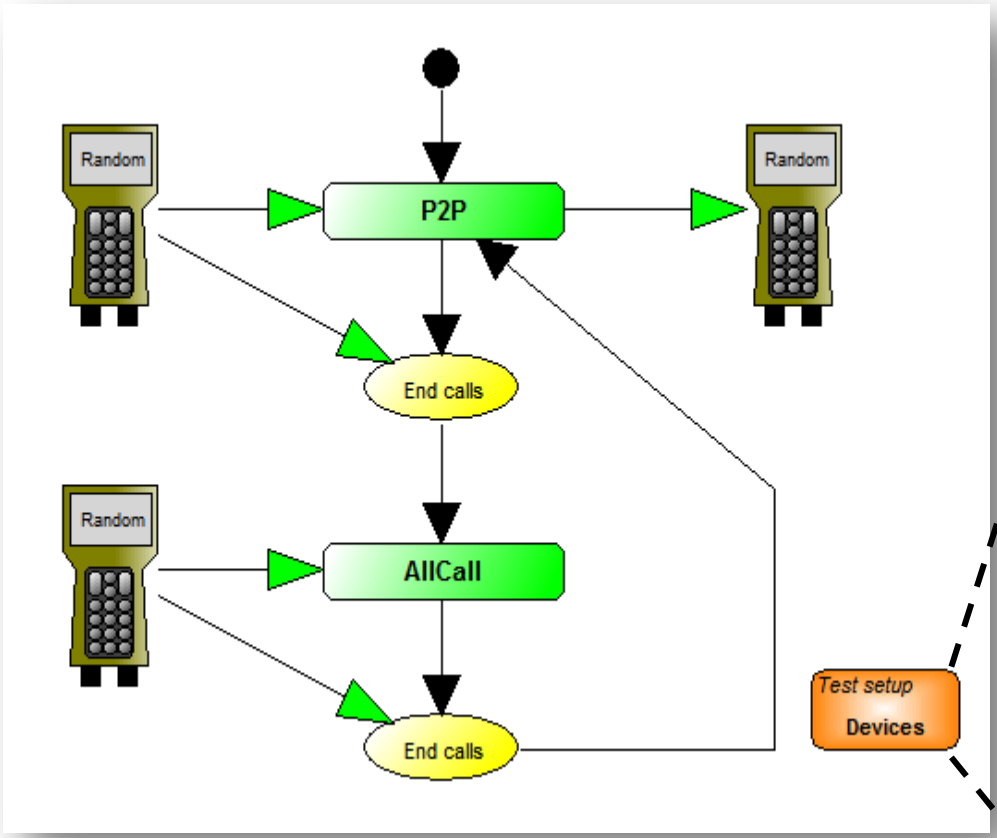
Language development

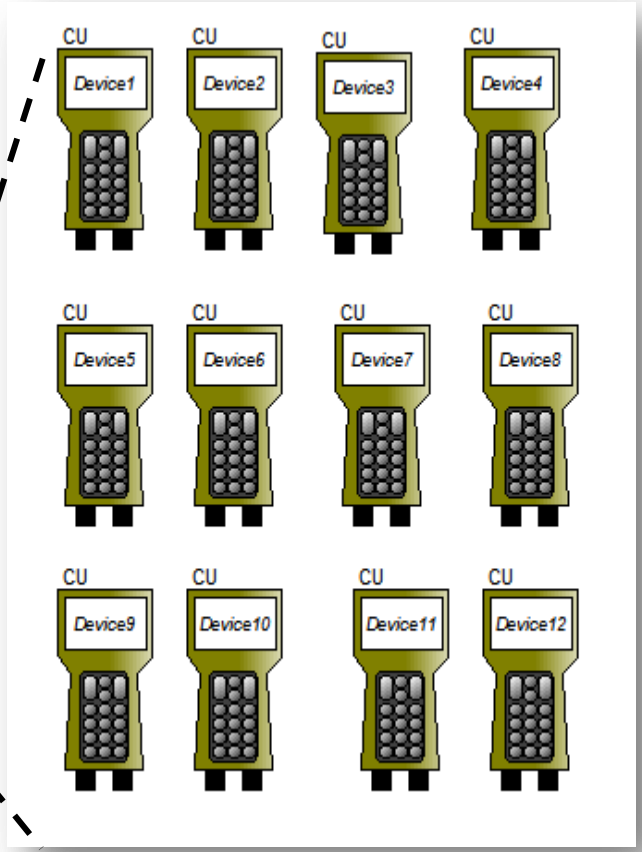


Model example 1: Modeling test cases



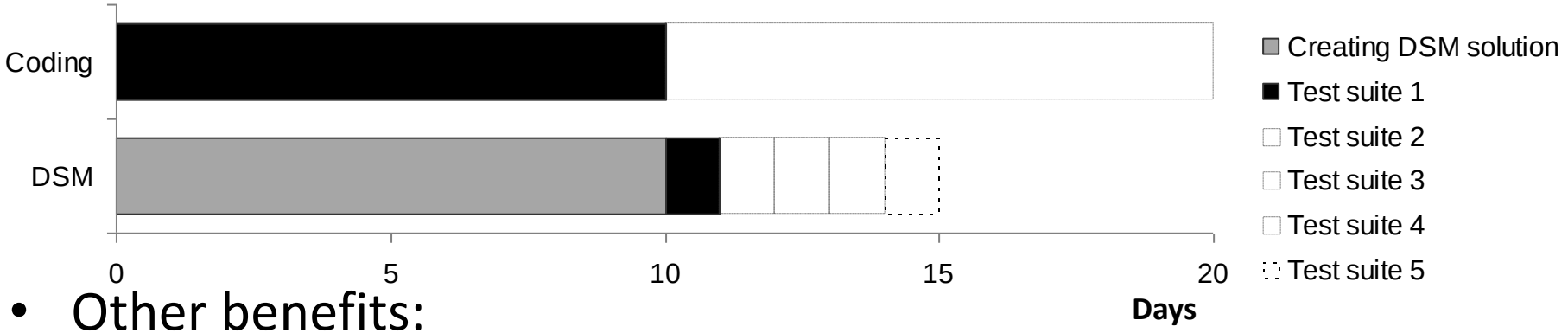
Model example 2: Modeling for test generation





Experiences

- About 10 times faster with modeling
- Set-up time estimation:
  - 2 weeks for the first version
  - 1 more week for making it better



- Other benefits:
  - Visualization makes it easy to understand
  - Easy test configuration
  - Test coverage dramatically increase with MBT
  - Mass testing with MBT models
  - No special skills needed for creating test cases

Results of combining DSLs + MBT

The case studies show:

- Easier adoption
  - Better acceptance, short ramp up
- Significantly faster model development
  - Higher abstraction leads to improved productivity
  - Automation of model creation
  - Immediate feedback & guidance during model creation
- Wider model accessibility
  - Visualization makes it easier to understand
  - Domain experts can participate
  - *Customers* can review models!

Summary

- Classic DSLs benefits found to be applicable in testing
  - Driven by fully automatic model transformations
  - Prevent illegal model construction & enforce methodology
- Challenge: Keep DSL lean *and* expressive
  - Leanness yields simplicity but too lean may lead to rejection!
  - Important to use tools that enable flexibility by allowing language evolution
- We believe DSL driven MBT will establish itself as the next step in evolution of software testing

How to get started on a DSL design

- Define
  - Concepts
  - Rules
  - Symbols
  - Generators
- Focus on how you think about a problem not how you (re)solve or describe it today
  - DSLs are not effective as graphical general purpose programming languages

How to get started: Concepts

- What are the different object types?
  - Example: Screen, forms widgets, messages
- What are their properties? What kind of values can they take? What is really relevant for testing?
  - Example: Dependencies between form fields? Yes
  - Example: Screen where button is located? Yes
  - Example: Pixel location of a button? No
  - Example: Underlying data base table structure? No
- What is the mapping domain concepts to concepts in the general purpose language?
  - Example: Button click maps to receiving a class

How to get started: Rules

- How many objects can exist?
  - Example: Only one starting point
- How can objects be connected?
  - Example: Only input actions can produce data
- Which property values have to be unique?
  - Example: Screen and form names
- What are valid property values?
  - Example: Only optional fields can be omitted
- When is a diagram ready for test generation?
  - Example: At least one input and verification action

How to get started: Symbols

- What type of diagrams are needed?
- Which objects are important to visualize in which diagram or at all?
  - Example: Author of a diagram
- What is the absolutely essential information important to get first understanding?
  - Example: Action has a pre-condition
- How should the information be represented?
  - Example: Symbol color, shape versus text

How to get started: Generators

- What type of information is needed to be generated?
  - Example: Code for test generation
  - Example: Model documentation
  - Example: “Live” model analysis
- In which order should objects be traversed to produce the generated code?
- How should property values be processed and converted to produce best target code?
- How to structure and modularize generator code to maximize reuse?

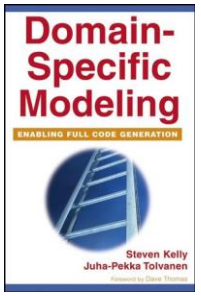
Thank you!

- Questions, comments, counter arguments, own experiences…

- Contact
  - Juha-Pekka Tolvanen [jpt@metacase.com]
  - www.metacase.com

  - Stephan Schulz [stephan.schulz@conformiq.com]
  - www.conformiq.com

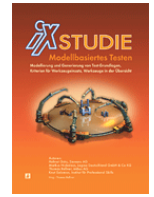
References [1/2]



- Kelly, S., Tolvanen, J.-P., Domain-Specific Modeling: Enabling Full Code Generation, Wiley, 2008. http://dsmbook.com
- Puolitaival, O.-P., et al, Utilizing Domain-Specific Modeling for Software Testing, Procs of VALID, October 2011
- Industrial presentations and tutorials at ETSI Conferences
  - http://www.model-based-testing.de/mbtuc11/program.html
  - http://www.elvior.com/model-based-testing-uc-2012/program

- MBT community http://model-based-testing.info/
- ETSI MBT Standardization
  - http://portal.etsi.org/portal/server.pt/community/MTS/323
  - MBT Modeling ES 202 951 http://pda.etsi.org/pda/queryform.asp

- *"Functional Testing Tools Are Not Enough."*, Forrester Research Inc. Report, Testing Tools Landscape, 2010
  - Summary available via www.conformiq.com
- „Modellbasiertes Testen [German only], iX Studie, 01/2009
  - Interesting (by now outdated) commercial MBT tool study from 2008



References [2/2]

- EADS, www.metacase.com/papers/MetaEdit_in_EADS.pdf
- NSN, Architecture in the language, www.metacase.com/cases/architectureDSMatNSN.html
- Nokia, www.metacase.com/papers/MetaEdit_in_Nokia.pdf
- Panasonic, Proceedings of Domain-Specific Modeling, 2007, www.dsmforum.org/events/DSM07/papers/safa.pdf
- Polar, Proceedings of Domain-Specific Modeling , 2009, www.dsmforum.org/events/DSM09/Papers/Karna.pdf
- USAF, ICSE, http://*dl.acm.org/citation.cfm?id=227842*