MetaCase

Choosing the Best Level of Abstraction for Your Domain-Specific Language

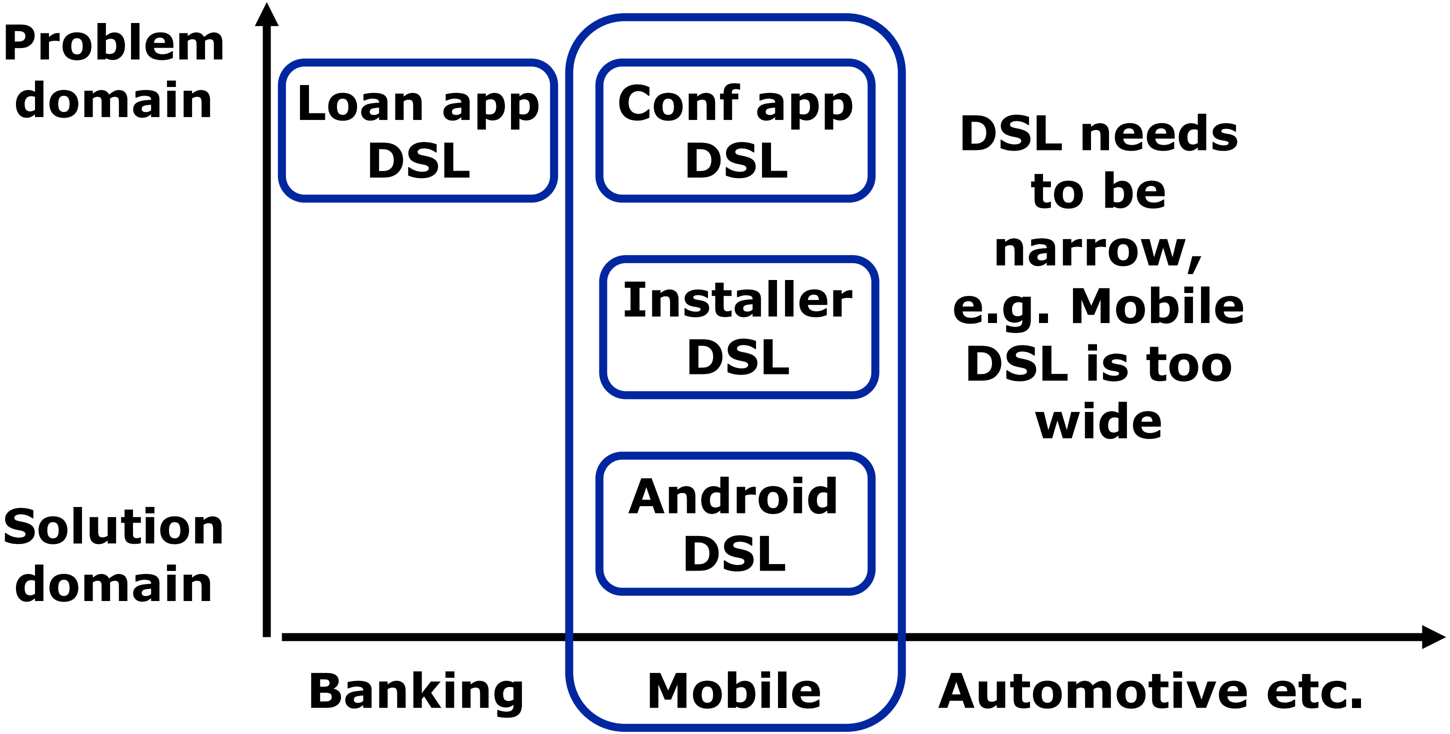
1 July 2013

Juha-Pekka Tolvanen, PhD
jpt@metacase.com

Some (repeated) definitions

- General-Purpose / Domain-Specific
- External / Embedded / Internal Languages
- Problem Domain / Solution Domain
- Graphical / text / matrix / table / maps etc. as concrete syntax
- Static / behavior
- Turing complete / in-complete
- Domain knowledge as first class citizen / naming convention

(Your) area of interest = domain



DSL needs to be narrow, e.g. Mobile DSL is too wide

Where at the main benefits?

- More empirical research is needed, but studies show improvements in:
  - Productivity
  - Quality
  - Process
  - Maintenance tasks
  - Understanding and communication
  - Easier introduction of new developers etc.
- While time-to-market is often the most significant improvement, many industry cases shows 5-10x (**500-1000%**) productivity improvements (see references)

Industry experiences

Elektrobit

"The setup effort to create the languages was a couple of weeks and provided more than **ten times faster speed**"

EADS

"The quality of the generated code is clearly better, simply because the modeling language **rules out errors**"

Nokia Siemens Networks

"The DSML solution makes development **significantly faster** and easier than the old manual coding practices"

Controlled empirical studies

Panasonic

- Built the same system twice: 425% faster
- Built code generator for a second platform: a fraction of time

POLAR®
LISTEN TO YOUR BODY

- Lab study: 6 engineers develop typical features: > 750% faster
- Built the same system twice: 900% faster

HOFERNET.com

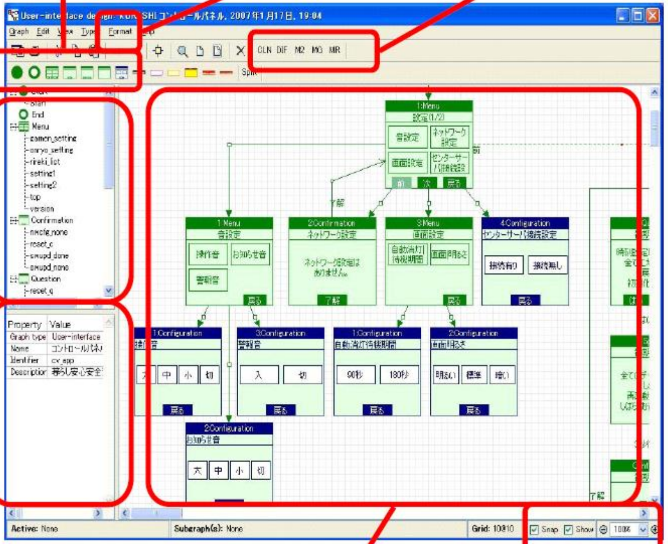
- Built the same production system in parallel
- Built several similar systems: Break Even Point = 3,14

Panasonic

Standard pages and connectors are available through the icon bar

On-site tool configuration facilities

Code generators and Work flow automations



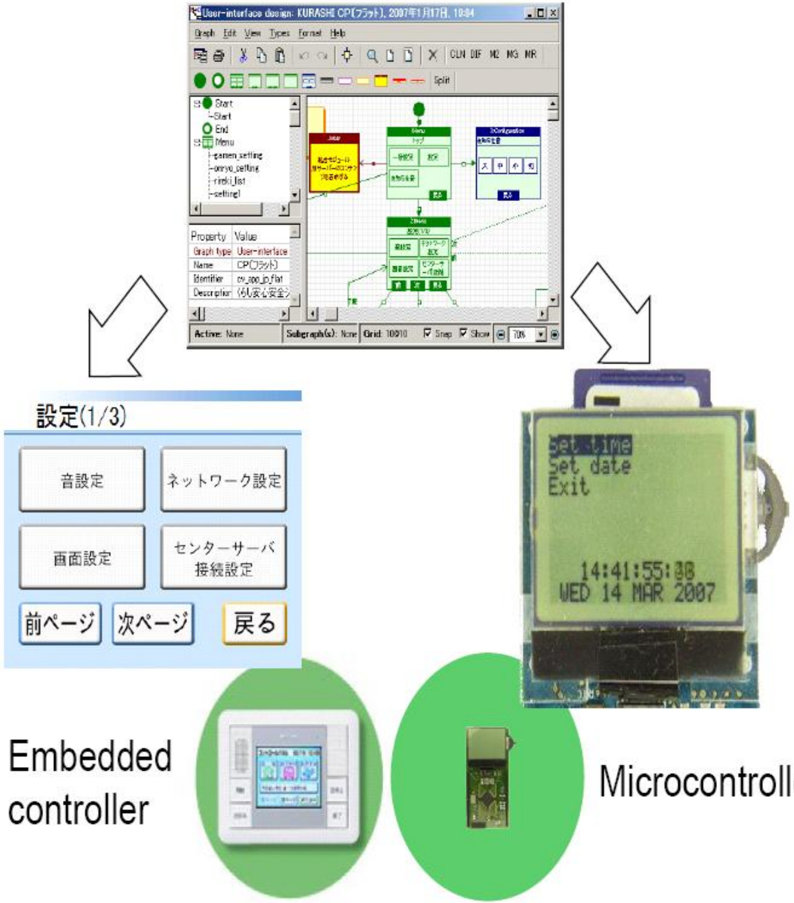
Property	Value
Graph type	User-interface
Name	コントロールパネル
Identifier	cv_app
Description	暮らし安心安全

Concepts browser

Detailed view

Visual modeling workspace

Zoom and other view modifiers

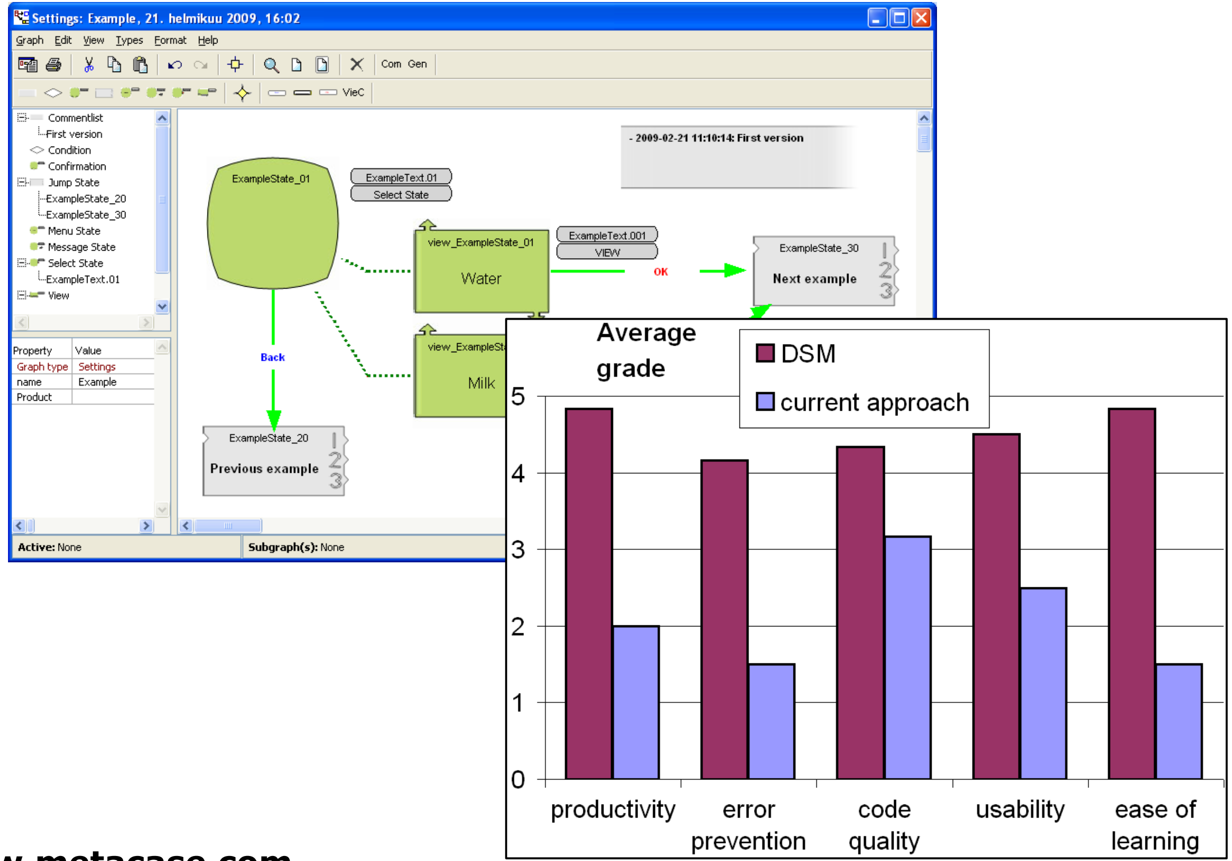


Embedded controller

Microcontroller

www.metacase.com

LISTEN TO YOUR BODY



Settings: Example, 21. helmikuu 2009, 16:02

- 2009-02-21 11:10:14: First version

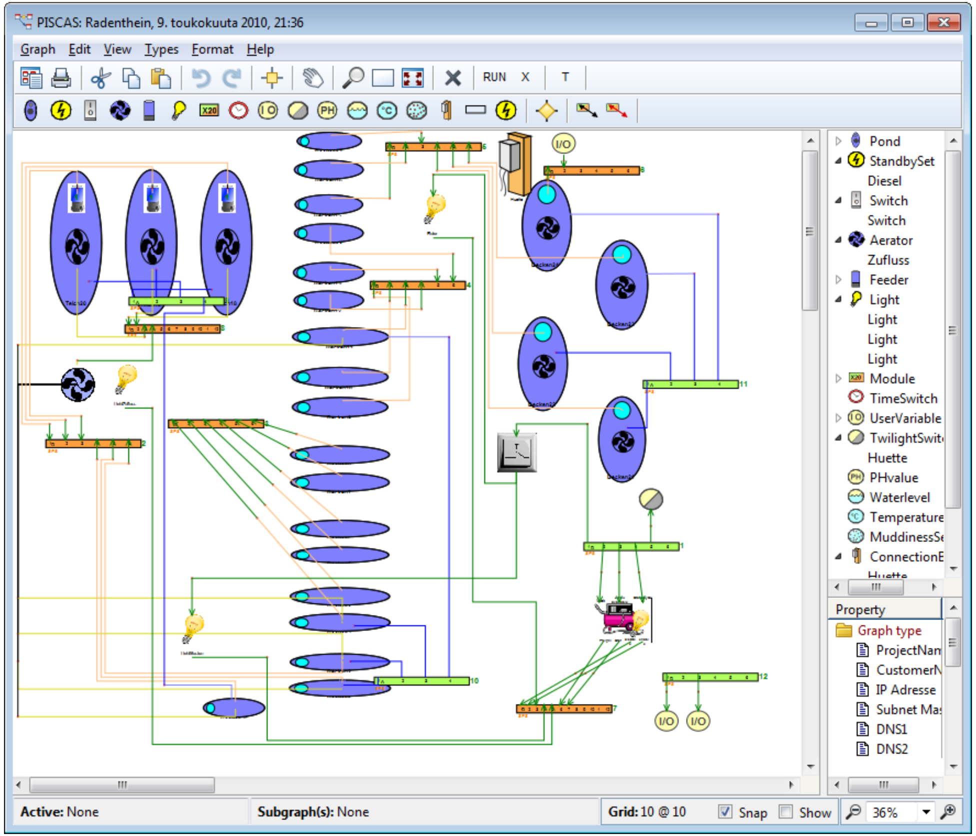
Property	Value
Graph type	Settings
name	Example
Product	

Active: None

Subgraph(s): None

www.metacase.com

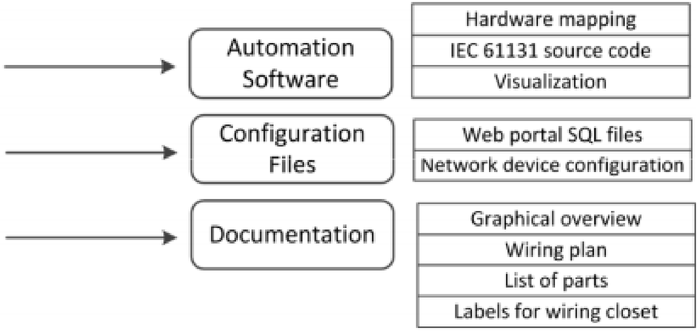
HOFERNET.com



Active: None

Subgraph(s): None

Grid: 10 @ 10 ☑ Snap □ Show 36%



Automation Software

- Hardware mapping
- IEC 61131 source code
- Visualization

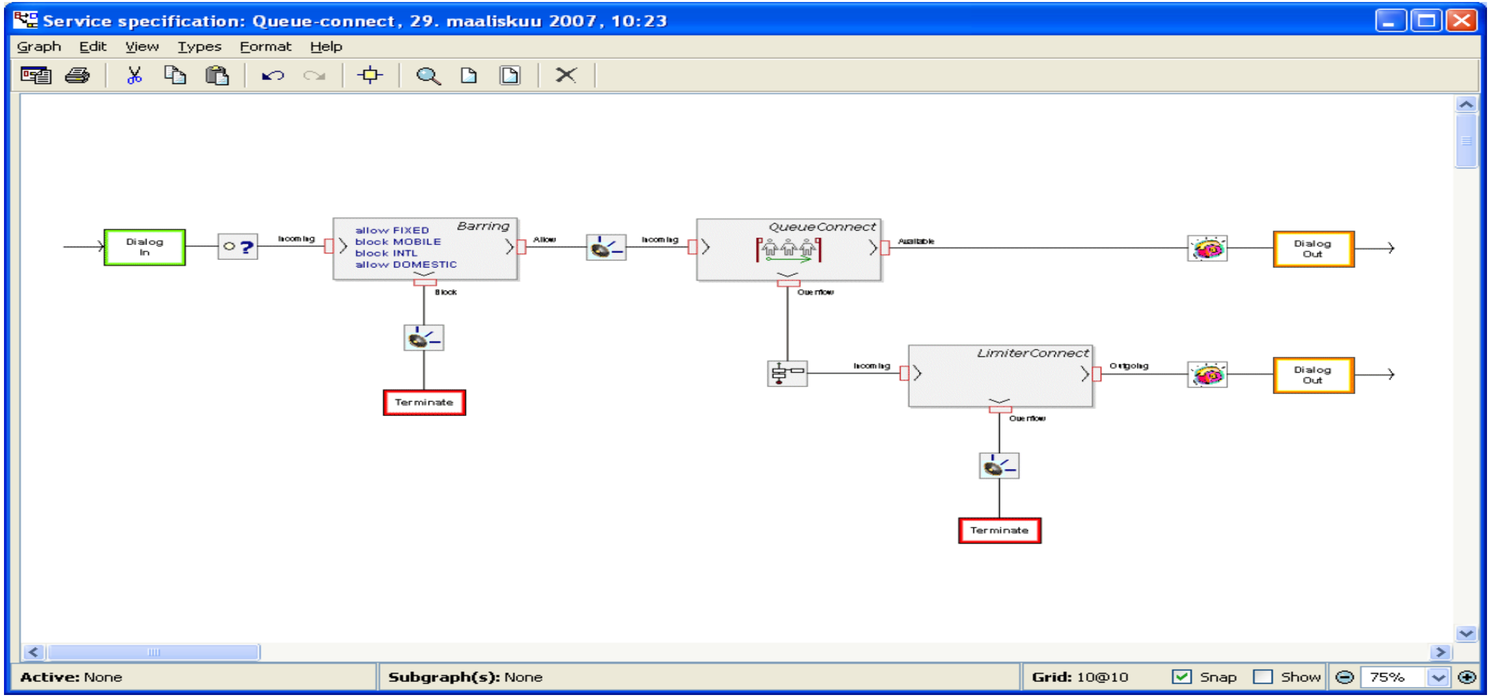
Configuration Files

- Web portal SQL files
- Network device configuration

Documentation

- Graphical overview
- Wiring plan
- List of parts
- Labels for wiring closet

Telecom services



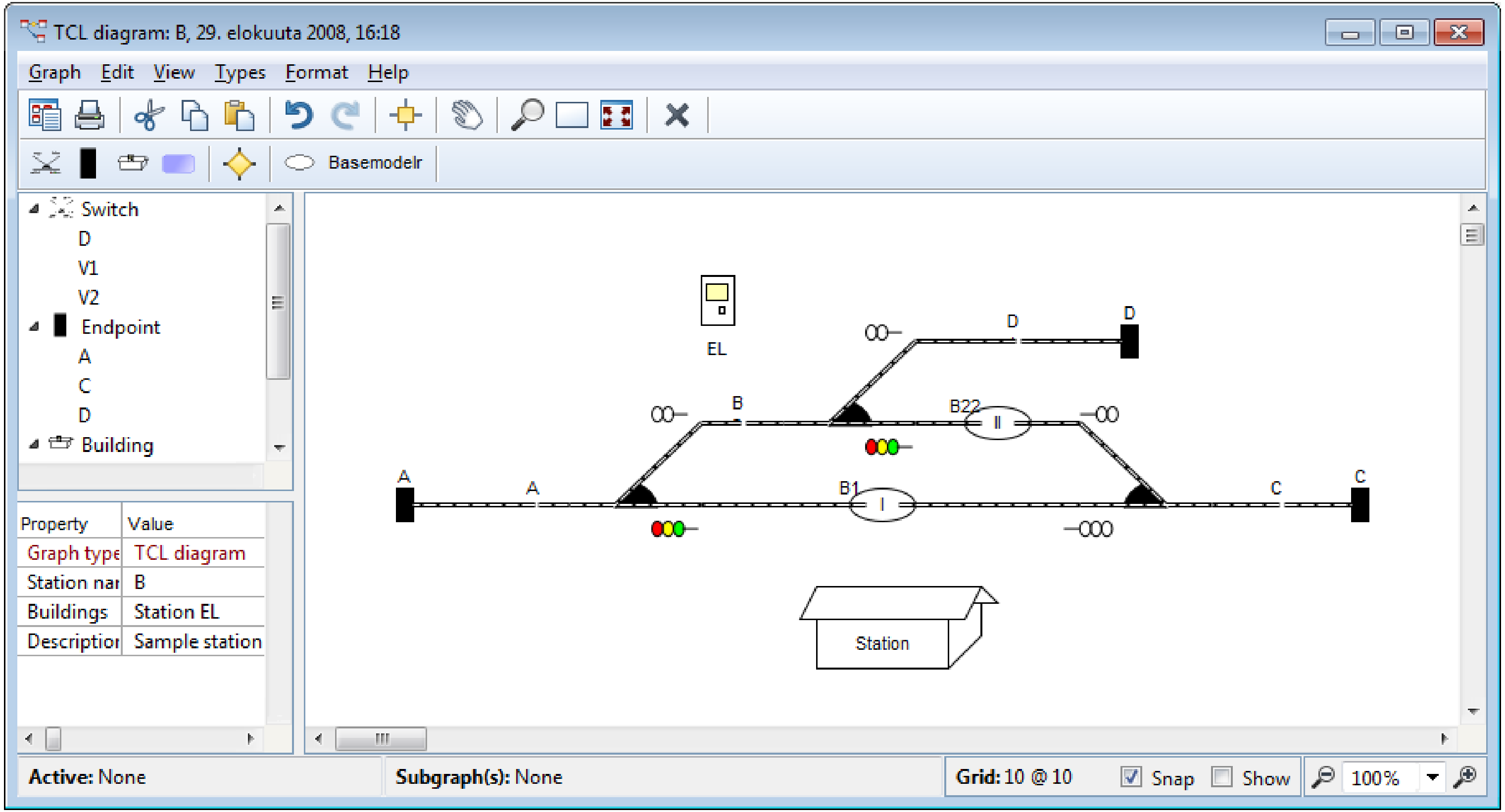
Service specification: Queue-connect, 29. maaliskuu 2007, 10:23

Active: None

Subgraph(s): None

Grid: 10@10 ☑ Snap □ Show 75%

Railway interlocking



TCL diagram: B, 29. elokuuta 2008, 16:18

- Switch
  - D
  - V1
  - V2
- Endpoint
  - A
  - C
  - D
- Building

Property	Value
Graph type	TCL diagram
Station nar	B
Buildings	Station EL
Descriptior	Sample station

Active: None

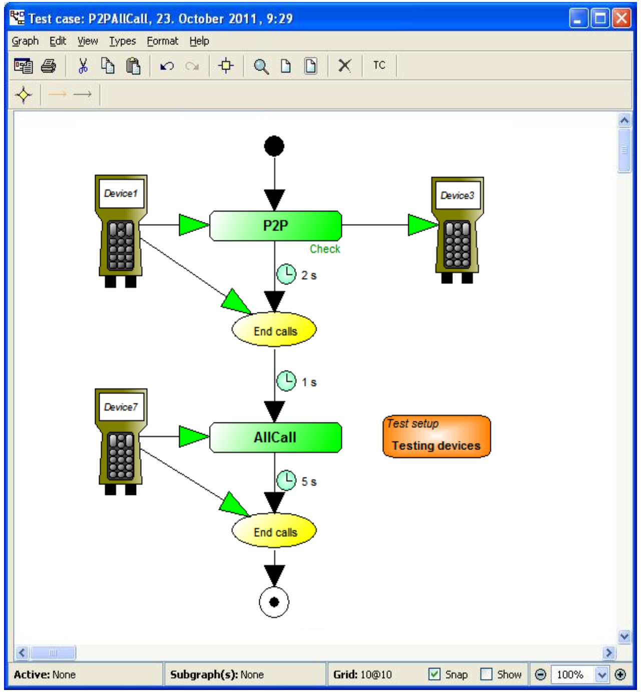
Subgraph(s): None

Grid: 10 @ 10 ☑ Snap □ Show

100%

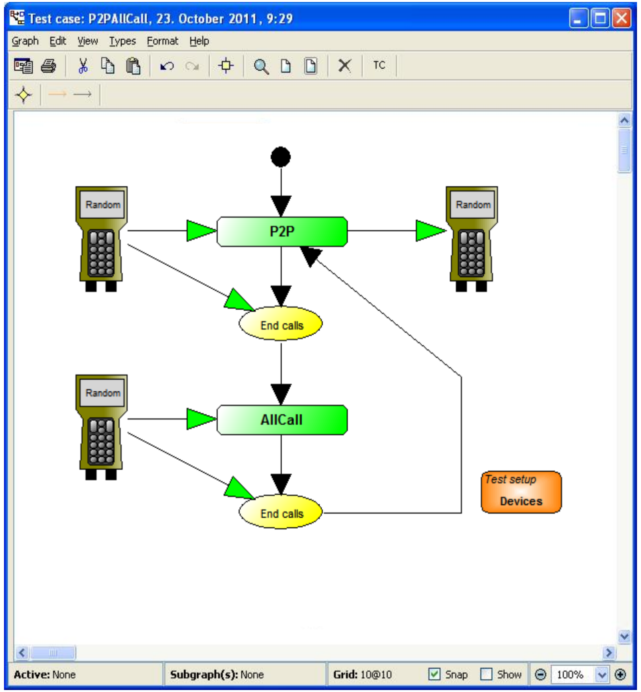
Radio network testing

- Modeling test cases/test logic and generating test data



Test case: P2PAllCall, 23. October 2011, 9:29

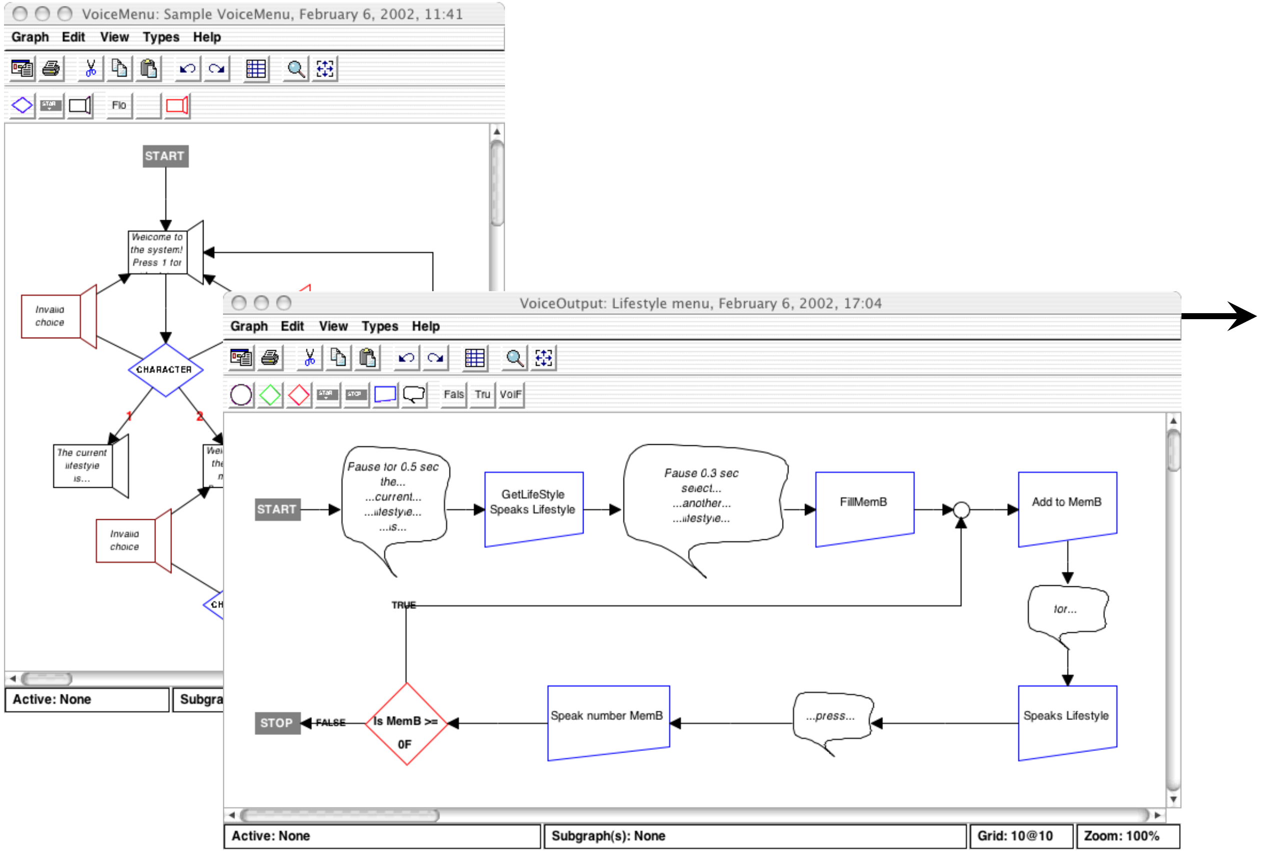
Active: None
Subgraph(s): None
Grid: 10@10
☑ Snap
□ Show
100%



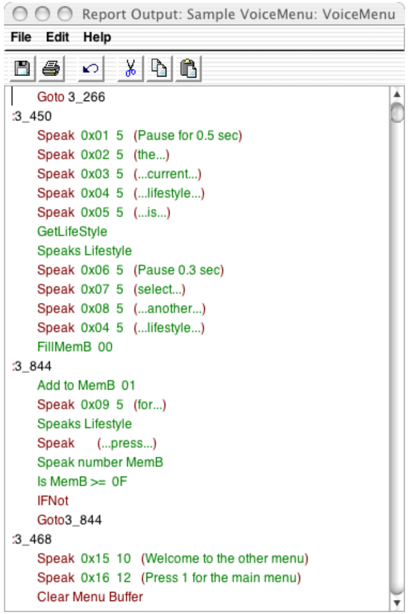
Test case: P2PAllCall, 23. October 2011, 9:29

Active: None
Subgraph(s): None
Grid: 10@10
☑ Snap
□ Show
100%

Voice control

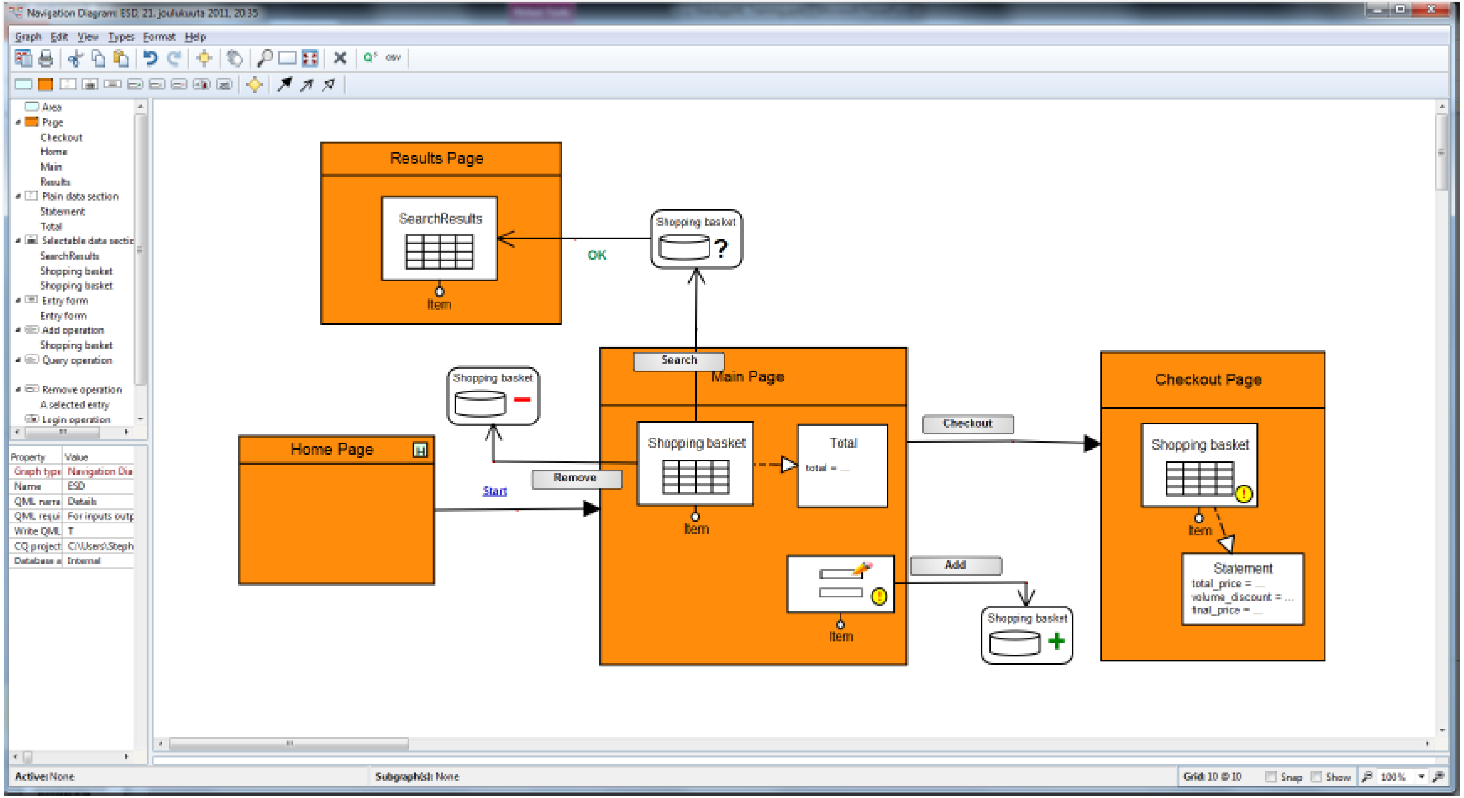


VoiceMenu: Sample VoiceMenu, February 6, 2002, 11:41
Graph Edit View Types Help
Flo
START
Welcome to the system! Press 1 for
Invalid choice
CHARACTER
1
2
The current lifestyle is...
Invalid choice
Active: None
Subgra
VoiceOutput: Lifestyle menu, February 6, 2002, 17:04
Graph Edit View Types Help
Fals Tru VoiF
START
Pause for 0.5 sec the... ...current... ...lifestyle... ...is...
GetLifeStyle Speaks Lifestyle
Pause 0.3 sec select... ...another... ...lifestyle...
FillMemB
Add to MemB
for...
Speaks Lifestyle
...press...
Speak number MemB
Is MemB >= 0F
TRUE
FALSE
STOP
Active: None
Subgraph(s): None
Grid: 10@10
Zoom: 100%



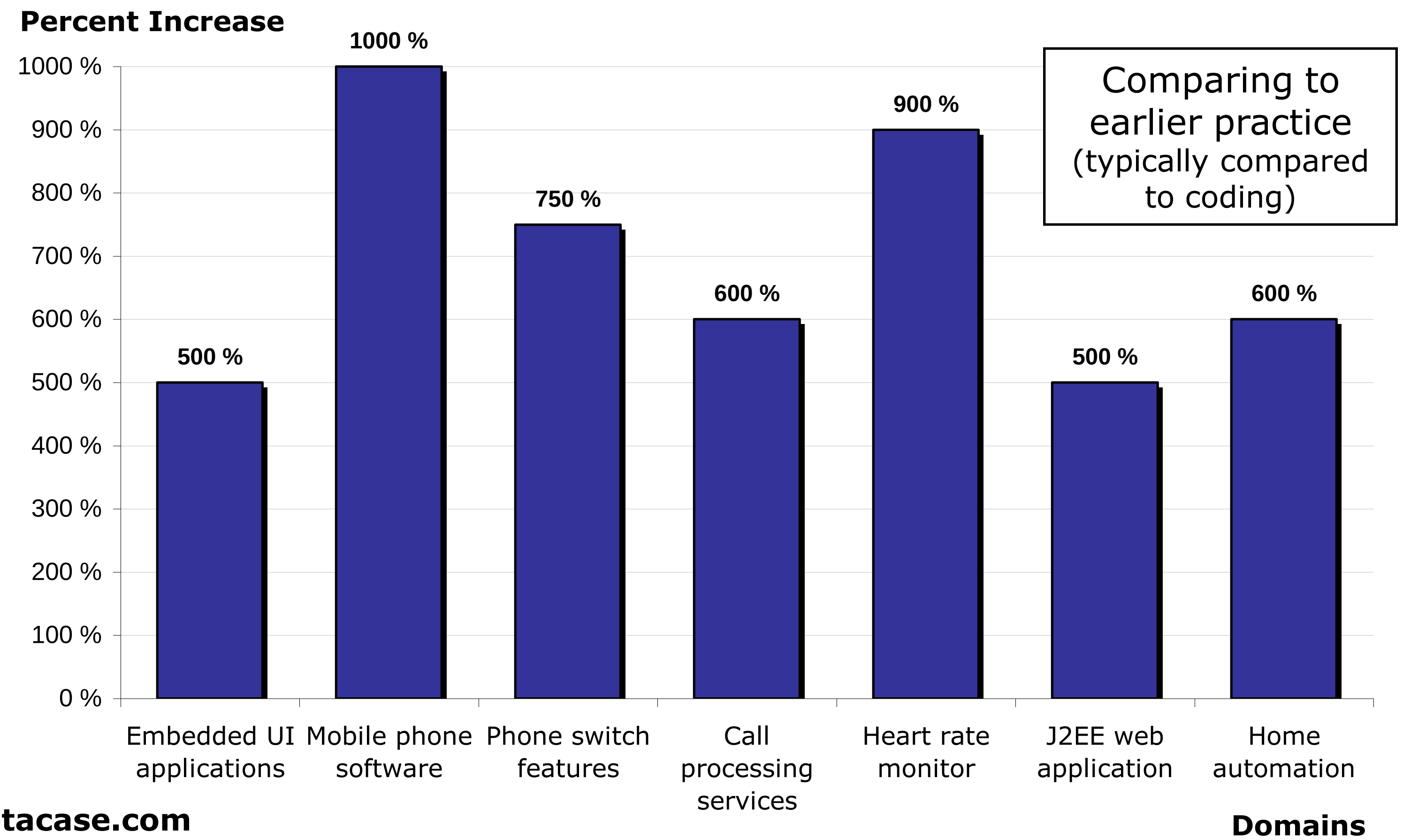
```
     Goto 3_266
:3_450
     Speak  0x01  5   (Pause for 0.5 sec)
     Speak  0x02  5   (the...)
     Speak  0x03  5   (...current...)
     Speak  0x04  5   (...lifestyle...)
     Speak  0x05  5   (...is...)
     GetLifeStyle
     Speaks Lifestyle
     Speak  0x06  5   (Pause 0.3 sec)
     Speak  0x07  5   (select...)
     Speak  0x08  5   (...another...)
     Speak  0x04  5   (...lifestyle...)
     FillMemB  00
:3_844
     Add to MemB  01
     Speak  0x09  5   (for...)
     Speaks Lifestyle
     Speak        (...press...)
     Speak number MemB
     Is MemB >=  0F
     IFNot
     Goto3_844
:3_468
     Speak  0x15  10   (Welcome to the other menu)
     Speak  0x16  12   (Press 1 for the main menu)
     Clear Menu Buffer
```

Web app testing



Property	Value
Graph type	Navigation Dia
Name	ESD
QML name	Details
QML requi	For inputs outp
Write QML	T
CQ project	C:\Users\Steph
Database a	Internal

Productivity increase measured



Comparing to earlier practice
(typically compared to coding)

Analysis of the cases indicates

- DSL should raise the level of abstraction above the code
- DSL is not using necessarily linear text representation
  - Domain and problem solving dictates
  - Maps, diagrams, matrixes, tables etc.
- DSL focus on narrow domain
  - Exclude outside as much as possible
- DSL for other than producing the code
  - Higher abstraction is not applicable only for producing the code but can also be used for testing, deployment, interaction design, localization etc.

Tooling

- Some sort of tooling is always used, but big differences on tools, see
  - series of Language Workbench Comparison, LWC 2011-2013
  - Comparison reports, e.g. tinyurl.com/gerard12: Language implementation (Eclipse GMF 25 days vs. MetaEdit+ 0.5 days)
- Language creation is the first task but also other issues need to be handled, like
  - integrating multiple languages
  - sharing languages
  - maintaining languages
  - updating specifications made with earlier version of language
  - collaboration, like multiple language engineers

Summary

- Raise the abstraction as high as possible: ideally 1:1 to problem domain
- Exclude outside as much as you can
- Refine languages as needed (keep it flexible)
- Use tools that support “agile” language definition
  - And allow also models to automatically update to new language version

MetaCase

Thank You!
Questions?

To see various cases, examples, and download MetaEdit+ tool, visit http://www.metacase.com

References

- Kelly, S., Tolvanen, J.-P., Domain-Specific Modeling: Enabling Full Code Generation, Wiley, 2008. DSMbook.com
- El Kouhen, A., Dumoulin, C., Gérard, S., Boulet, P., Evaluation of Modeling Tools Adaptation, tinyurl.com/gerard12
- Kärnä, J., et al. Evaluating the Use of Domain-Specific Modeling in Practice, 9th DSM Workshop (2009)
- Puolitaival, et al. Utilizing Domain-Specific Modeling for Software Testing, Proceedings of VALID, (2011)
- Preschern et al. Domain Specific Language Architecture for Automation Systems: An Industrial Case Study, Procs of Graphical Modeling Language Development, DTU (2012)
- Safa, L., The Making Of User-Interface Designer, A Proprietary DSM Tool, Procs of 7th DSM Workshop (2007)
- Sprinkle et al. (eds) IEEE Software, DSL&M special issue, July/Aug, 2009, including: Kelly & Pohjonen, Worst Practices for DSM, tinyurl.com/worstDSM