DSLs and Model-Based Approach in Product Line Development

PhD, Prof. Dmitry Koznov
Saint-Petersburg State University, Russia

30 May, 2023

Agenda

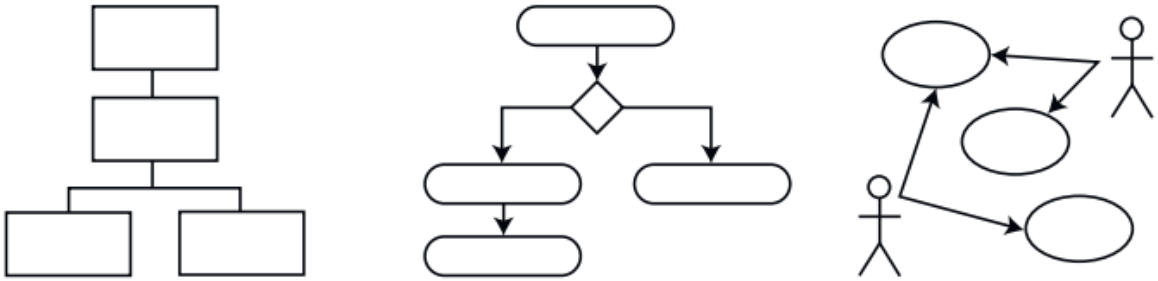
- Model-Based Approach
- DSLs
- Product Line approach
- Example
- Conclusions and Discussion

Agenda

- Model-Based Approach
- DSLs
- Product Line approach
- Example
- Conclusions and Discussion

Model-Based Approach

- This is a paradigm of software development (mode-based development, visual modeling), which:
  - Focusing on designing of well-defined abstractions
  - Supporting multiple view-points on a system being developed
  - Using graph models for software presentation (visualization)
  - May be applied for various development activities (requirement management, software design, maintenance, documentation, communications, etc.) *)



*) *D.Koznov. Bases of Model-Based Approach. Binom, 2008*

Model-Based Approach

Analogy with construction

- A small number of highly cost experts

- A big number of low cost developers

Brief History of Model-Based Approach

- XVIII в., Gaspard Monge,mathematica bases of descriptive geometry (Géométrie descriptive)
- XIX в., wide-spreading technical drawing in engineering
- 1947 – John von Neumann suggested flowcharts (Planning and coding problems for an electronic computing instrument)
- 50s – flowchart standards (IBM, ANSI)
- 60s – methods for design of artificial systems
- 70s – structured analysis methods (SADT, etc.)
- 70-80s design languages for telecom systems, CCITT/ ITU (SDL, MSC)
- 90s – more than 50 object-oriented analysis & design methodologies
- 1997 г. – Unified Modeling Language (UML)
- 2000s modeling standards: SysML, BPMN, MDA, etc.
- 2000-2010s – domain-specific modeling

But something was wrong….

Two papers

- Flowchart are ineffective, 1977
  - Shneiderman, B. Experimental Investigations of the Utility of Detailed Flowcharts in Programming / B. Shneiderman, R. Mayer, D. McKay, P. Heller // Communications of the ACM. — 1977. — 20 (6). — P. 373–381.
- UML doesn't use in industry, 2013
  - Petre, M. UML in Practice / M. Petre // Proc. of 35th International Conference on Software Engineering (ICSE). — 2013. — P. 722–731.

Agenda

- Model-Based Approach
- DSLs
- Product Line approach
- Example
- Conclusions and Discussion

Agenda

- Model-Based Approach
- DSLs
- Product Line approach
- Example
- Conclusions and Discussion

Domain-Specific Languages (DSLs)

- Data of birth is unknow….
- Project languages basing on macros in COBOL, PL/1, C++ (70-2000)*)
- Wide-spreading XML and appearing a famous book of M.Fawler (2011) **)
- Tool support of DSLs – MPS, Eclipse/xtext, etc. (2010 – up to now)

*) *D. Yu. Boulychev, D. V. Koznov, A. A. Terekhov. On Project-Specific Languages and Their Application in Reengineering. CSMR 2002: 177-185*

**) *M. Fowler. Domain-Specific Languages. Williams Publishing House, 2011*

Agenda

- Model-Based Approach
- DSLs
- Product Line approach
- Example
- Conclusions and Discussion

Agenda

- Model-Based Approach
- DSLs
- Product Line approach
- Example
- Conclusions and Discussion

Product Line Examples

SUVs of Toyoda

FJ Cruiser

Land Cruiser

Hilux

Fortuner

Tacoma

Lenovo ThinkPads

E15, 15.6“, AMD Ryzen 3 5300U

P15, 15.6“, Intel Core i7, NVIDIA

T14, 14, Intel Core i5

IP Gaming 3, 15.6“,NVIDIA, Intel Core i7

ThinkBook 15 G3, 15.6“, AMD 5300U

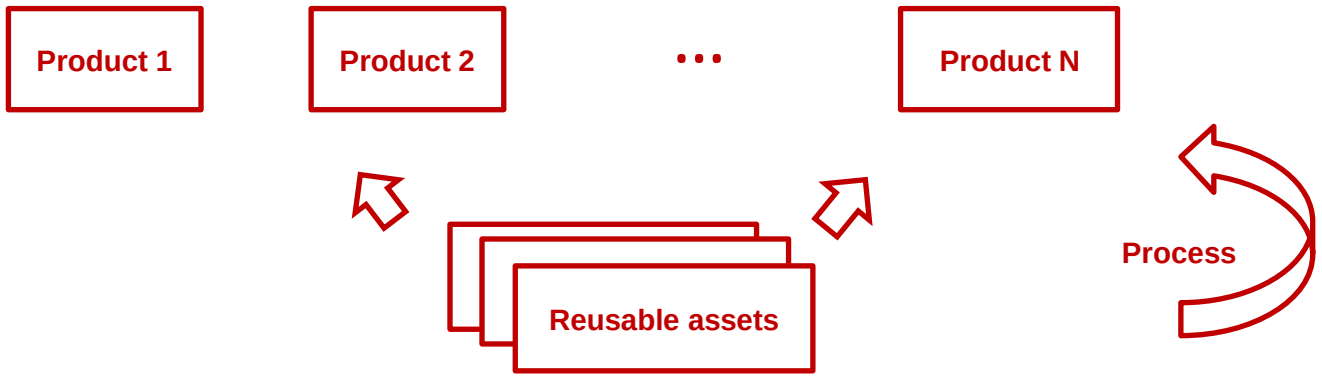
ThinkPad X1 Yoga Gen 6, 14“, Intel Core i7

Microsoft OS Windows Familia

- Windows 11
- Windows 10 Mobile
- Windows Server 2022
- Windows Embedded Server

Product Line Definition

- A software product line is a set of software-intensive systems that share a **common**, managed **set of features** satisfying the specific needs of a particular market segment or mission and that are developed from a **common set of core assets in a prescribed way**[*)]



*) *A Framework for Software Product Line Practice, version 5.0, Software Engineering Institute ©*

Reuse in Product Lines

Product 1

Product 2

Product 3

Reusable assets

Reusable assets

- Source code (components, classes, interfaces, etc.)
- Architecture and infrastructure
- Tests, test scenarios, test data
- Requirements
- Models and specifications
- Documentation

Software Product Lines

Current research state

- More than 4000 research papers in 1990 -2021
- A Framework for Software Product Line Practice, version 5.0, Software Engineering Institute ©, 2012
- Software Product Line Conference (Rank A)
- However right now the topic is not included in high list of leading software engineering conferences (ICSE, ASE)
- Actually, Product Lines are rather a matter of practice than a subject of research at the moment

Software Product Lines

Current research state

- More than 4000 research papers in 1990 -2021
- A Framework for Software Product Line Practice, version 5.0, Software Engineering Institute ©, 2012
- Software Product Line Conference (Rank A)
- However right now the topic is not included in high list of leading software engineering conferences (ICSE, ASE)
- **Actually, Product Lines are rather a matter of practice than a subject of research at the moment**

Telecom Product Lines

Challenges

- **Management of product configurations**
  - A big number of products
  - A considerable number of supported interfaces
  - Various hardware combinations for the same product
  - Finally, a huge number of xml configuration files
- **Reuse** of assets for development of new products
  - Code
  - Data & configurations
  - Specifications, etc.
- **Quality assurance** of final products
  - Code analysis
  - Test generation
  - Simulation and debugging
  - Quality assurance on cross-development layers
- **Decouple** of layers and components
- **A huge volume resources** is involved to development

Product Lines and XML

- Various kinds of configuration
  - Products
  - Interfaces
  - Data, etc.
- I found > 10 kinds of XML in a one telecom product line
- There are thousands lines of XML code
- This code is hard for analyzing and modification
- It is really up-to-date for telecommunications

```
<class opright="query" type="CFG" name="DrvLoadCmdDef">
    <defaultrecord>
        <record domain="LR|PR|VR" opright="add|del|modify|query">
            <attr name="moduleType" value="PICS5732_H24S4X6QZ"/>
            <attr name="cmdID" value="1"/>
            <attr name="cmdType" value="CMD_TYPE_SO"/>
            <attr name="driverName" value="libvpic"/>
            <attr name="rpmCptName" value="libvpic.so"/>
            <attr name="CondID" value="0"/>
            <attr name="failDisposeId" value="0"/>
            <attr name="loadMode" value="LOAD_MODE_SYNC"/>
        </record>
        <record domain="LR|PR|VR" opright="add|del|modify|query">
            <attr name="moduleType" value="PICS5732_H24S4X6QZ"/>
            <attr name="cmdID" value="2"/>
            <attr name="cmdType" value="CMD_TYPE_KO"/>
            <attr name="driverName" value="vpic"/>
            <attr name="rpmCptName" value="vpic_8572.ko"/>
            <attr name="CondID" value="0"/>
            <attr name="failDisposeId" value="0"/>
            <attr name="loadMode" value="LOAD_MODE_SYNC"/>
        </record>
    </defaultrecord>
</class>
```

```
<class opright="query" type="CFG" name="ModuleCompDef">
        <defaultrecord>
            <record domain="LR|PR|VR" opright="add|del|modify|query">
                <attr name="moduleType" value="PICS5732_H24S4X6QZ"/>
                <attr name="devName" value="vpiccard@slotid@"/>
                <attr name="hardType" value="PICS5732_H24S4X6QZ"/>
                <attr name="posAttrID" value="SERIAL_NO"/>
                <attr name="posAttrValue" value="4294967295"/>
                <attr name="isContainer" value="0"/>
                <attr name="needPoll" value="0"/>
                <attr name="seqNum" value="0"/>
                <attr name="showName" value="vpiccard@position@"/>
                <attr name="channelType" value="0"/>
                <attr name="role" value="1"/>
            </record>
    </defaultrecord>
</class>
<class opright="query" type="CFG" name="ModuleModelDef">
        <defaultrecord>
            <record domain="LR|PR|VR" opright="add|del|modify|query">
                <attr name="moduleType" value="PICS5732_H24S4X6QZ"/>
                <attr name="devName" value="vpiccard@slotid@"/>
                <attr name="modelType" value="0"/>
                <attr name="parentDevName" value="card_slot@slotid@"/>
            </record>
    </defaultrecord>
</class>
```

Agenda

- Model-Based Approach
- DSLs
- Product Line approach
- Example
- Conclusions and Discussion

Agenda

- Model-Based Approach
- DSLs
- Product Line approach
- Example
- Conclusions and Discussion

Switch & Router

- **Switch** is a networking device that connects other devices on a computer network
- **Router** – is a networking device that forwards data packets between computer networks. In particular, routers perform the traffic directing functions between networks and on the global Internet

Switch Huawei S5735-L24T4S-A1

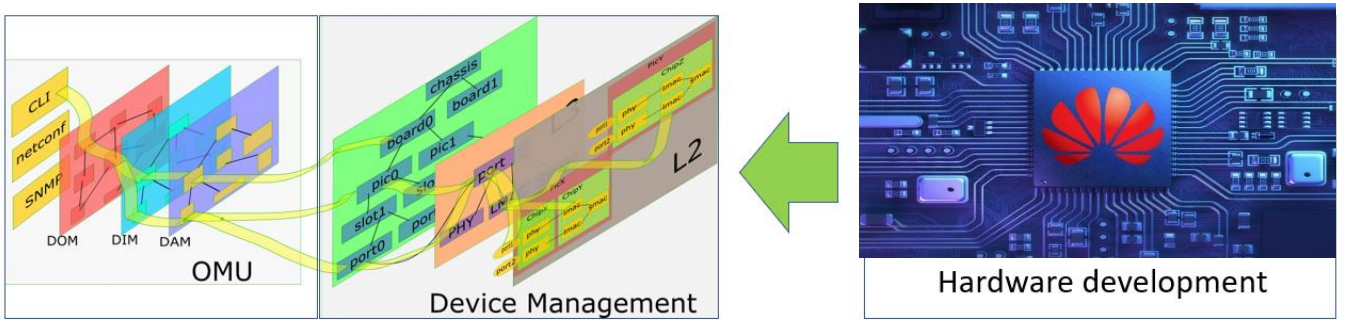
Router Huawei WS550

Device Management

A subdomain of switch/router product line

- It is a broad term that includes various administrative tools and processes for the maintenance of a computing, network, mobile and/or virtual device
- In our context, Device Management is the abstraction layer above device's drivers which provides unified managing interface for OAM across the whole product line

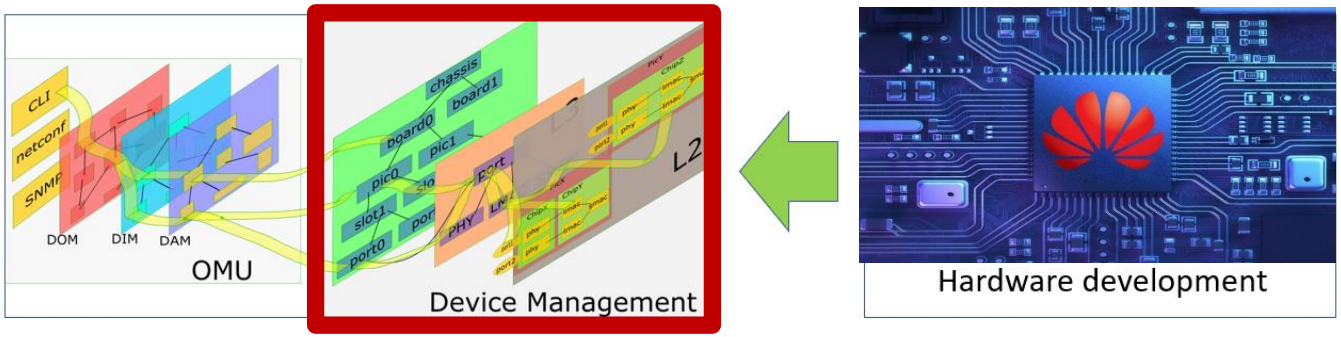
Switch/Router Product Line



- More than 40 various products
- Around 10 millions of code lines
- Use various programming languages including XML

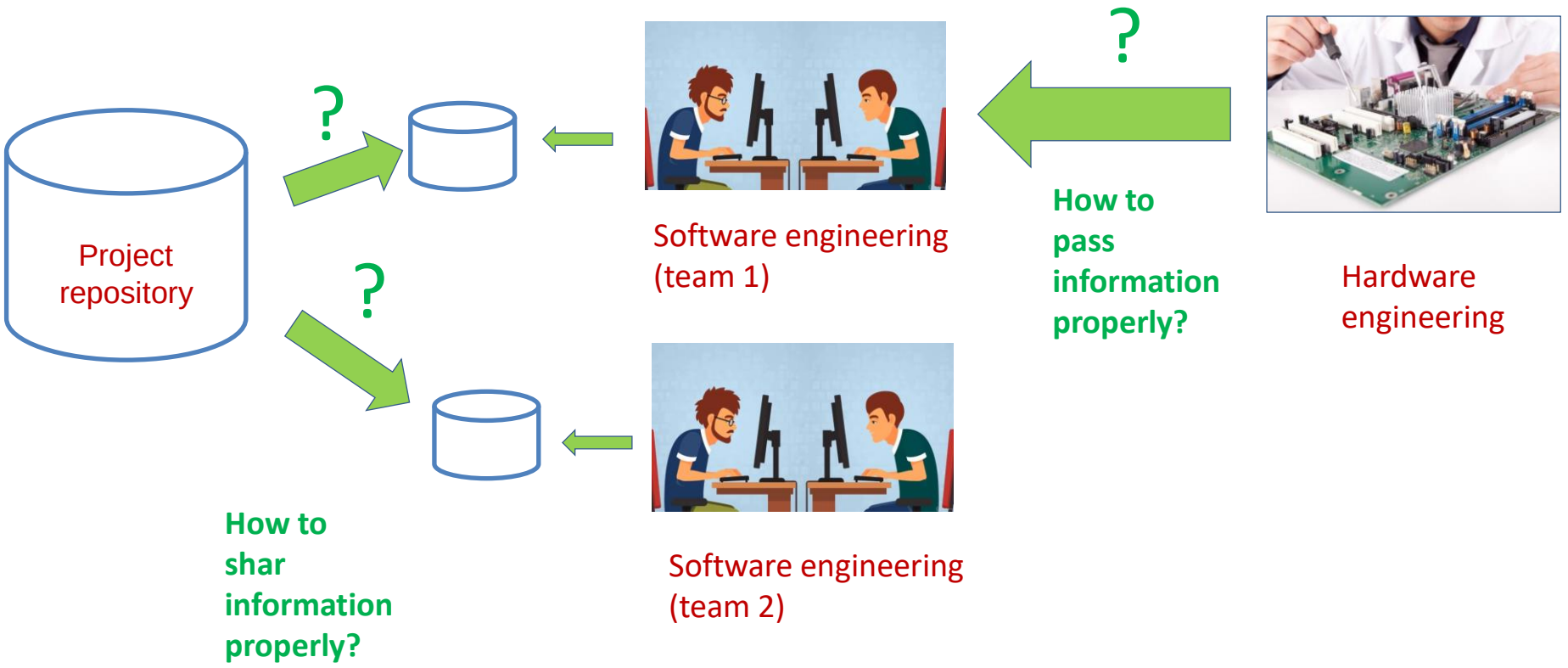
Switch/Router Product Line

Challenges



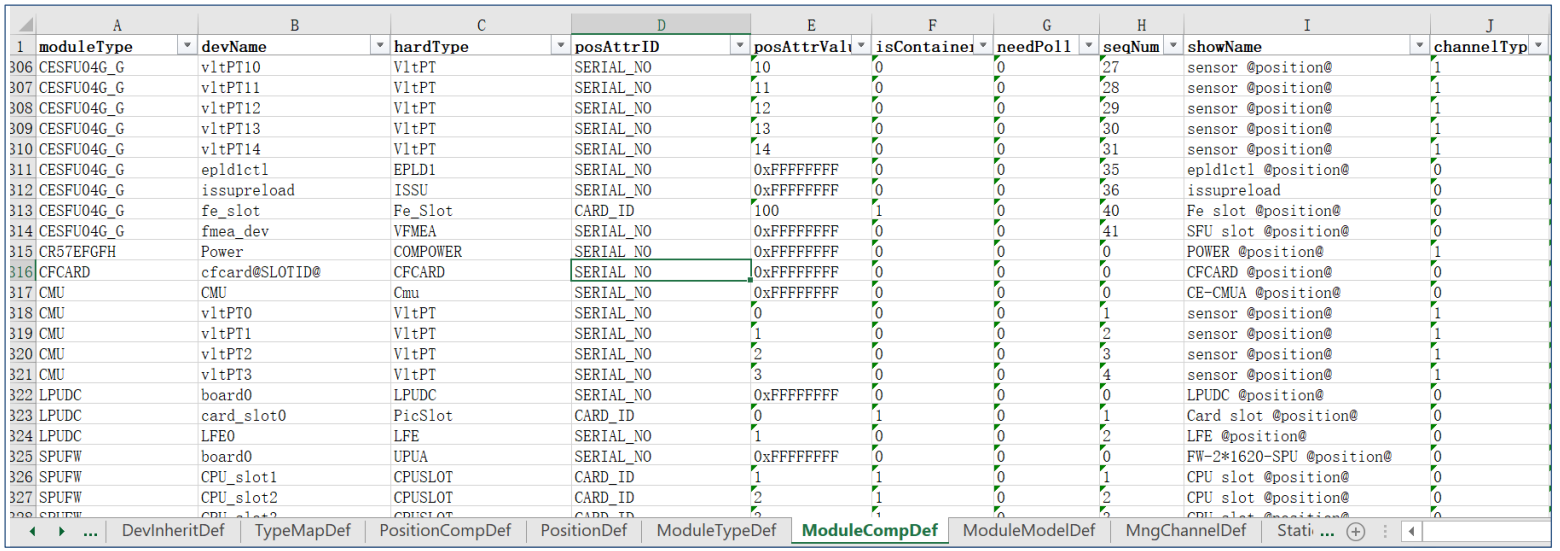
- **Problem 1. Difficult in information sharing** between various experts and development teams
- **Problem 2. Reuse issues** when new product being created, in particular, **error prone process** of product **configuration**

Problem 1. Difficult in Information Sharing



Problem 2. Reuse Issues

- **A huge volume of partially structured information** about products is specified in XML/Excel formats (a lot of small pieces of information, complicated structure, legacy information, etc.)
- As a result, **domain abstractions are split** into huge number attributes/tags/ rows and columns
- Consequently, **considerable efforts** are spent **to reuse** this information when new products are created

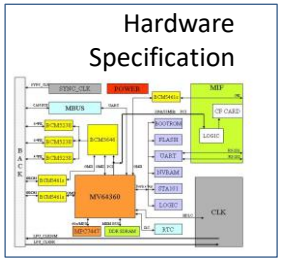


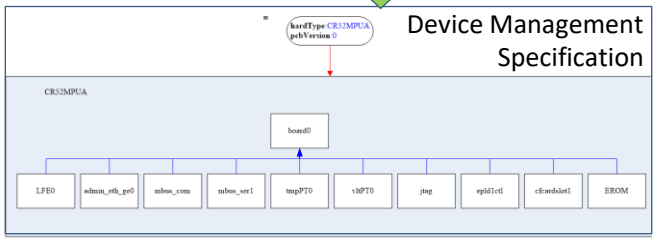
	A	B	C	D	E	F	G	H	I	J
1	moduleType	devName	hardType	posAttrID	posAttrVal	isContaine	needPoll	seqNum	showName	channelTyp
306	CESFU04G_G	vltPT10	VltPT	SERIAL_NO	10	0	0	27	sensor @position@	1
307	CESFU04G_G	vltPT11	VltPT	SERIAL_NO	11	0	0	28	sensor @position@	1
308	CESFU04G_G	vltPT12	VltPT	SERIAL_NO	12	0	0	29	sensor @position@	1
309	CESFU04G_G	vltPT13	VltPT	SERIAL_NO	13	0	0	30	sensor @position@	1
310	CESFU04G_G	vltPT14	VltPT	SERIAL_NO	14	0	0	31	sensor @position@	1
311	CESFU04G_G	epld1ctl	EPLD1	SERIAL_NO	0xFFFFFFFF	0	0	35	epld1ctl @position@	0
312	CESFU04G_G	issupreload	ISSU	SERIAL_NO	0xFFFFFFFF	0	0	36	issupreload	0
313	CESFU04G_G	fe_slot	Fe_Slot	CARD_ID	100	1	0	40	Fe slot @position@	0
314	CESFU04G_G	fmea_dev	VFMEA	SERIAL_NO	0xFFFFFFFF	0	0	41	SFU slot @position@	0
315	CR57EFGFH	Power	COMPOWER	SERIAL_NO	0xFFFFFFFF	0	0	0	POWER @position@	1
316	CFCARD	cfcard@SLOTID@	CFCARD	SERIAL_NO	0xFFFFFFFF	0	0	0	CFCARD @position@	0
317	CMU	CMU	Cmu	SERIAL_NO	0xFFFFFFFF	0	0	0	CE-CMUA @position@	0
318	CMU	vltPT0	VltPT	SERIAL_NO	0	0	0	1	sensor @position@	1
319	CMU	vltPT1	VltPT	SERIAL_NO	1	0	0	2	sensor @position@	1
320	CMU	vltPT2	VltPT	SERIAL_NO	2	0	0	3	sensor @position@	1
321	CMU	vltPT3	VltPT	SERIAL_NO	3	0	0	4	sensor @position@	1
322	LPUDC	board0	LPUDC	SERIAL_NO	0xFFFFFFFF	0	0	0	LPUDC @position@	0
323	LPUDC	card_slot0	PicSlot	CARD_ID	0	1	0	1	Card slot @position@	0
324	LPUDC	LFE0	LFE	SERIAL_NO	1	0	0	2	LFE @position@	0
325	SPUFW	board0	UPUA	SERIAL_NO	0xFFFFFFFF	0	0	0	FW-2*1620-SPU @position@	0
326	SPUFW	CPU_slot1	CPUSLOT	CARD_ID	1	1	0	1	CPU slot @position@	0
327	SPUFW	CPU_slot2	CPUSLOT	CARD_ID	2	1	0	2	CPU slot @position@	0

DevInheritDef | TypeMapDef | PositionCompDef | PositionDef | ModuleTypeDef | **ModuleCompDef** | ModuleModelDef | MngChannelDef | Stati ...

Thus, new abstractions are needed!

Specification of a board/card: a lot of details





Device Management view point on the board: just some details

- Introducing new device management entities and building an special hierarchies for them
  - + Structured driver related information
  - + Code-related viewpoints on hardware functionality
  - + Code-related viewpoints on target device external interfaces
  - + Data path specification, etc.

Device Management DSL (DevM)

- **Textual C-like** domain specific language
- **Separation of concerns** paradigm (multiple view supporting)
- **Visualization tools** for navigation on specification structure and data paths
- **Development framework** integrated into target environment
- **Smart generation** of target code
- **Reverse engineering facilities**

DevM

Key abstractions

- Boards/cards include
  - Chips
  - Panelports & ports
  - Bus
  - Slots/toslots
  - Datapath description
  - Drivers loading specifications
- Optical modules can be inserted into panelports to split them

```
Type AAAA_L1: BOARD {
 devType = PhyMainBrd
override ATTR_I deployOS = 1}

BOARD AAAA_L1 aaaa_l1 (declare position) (
      devName = board0, .....)
{
   CHIP LFE LFE0 (
      showName = LFE @position@,
      devName = LFE0,
      posAttrID = SERIAL_NO,
      posAttrValue = 1,
   );

   SLOT PicSlot card_slot0 (
      showName = card slot @position@,
      devName = card_slot0,
      posAttrID = CARD_ID,
      posAttrValue = 0,
      isContainer = 1,
   );

   PORT EthMGEPort admin_eth_ge0 (
      showName = MEth@position@,
      devName = admin_eth_ge0,
      posAttrID = PORT_ID,
      posAttrValue = 1    );
```

DevM Domain Specific Language

Viewpoints supported

- **Optical model** – representation of optical module capabilities and properties
- **Panelport model** – description of data path related to certain panelport from driver point of view
- **Composition model** – description of available hardware components of target device for providing network services from node administrator point of view
- **Inheritance model** – available services hierarchy provided by different device types
- **Behavior model** – specification of reactions for standard events reported by different hardware components of target device

Simplified Composition Model for a Board

```
Type AAAA_L1: BOARD {
 devType = PhyMainBrd
override ATTR_I deployOS = 1}

BOARD AAAA_L1 aaaa_l1 (declare
position) (
      devName = board0, .....)
{
   CHIP LFE LFE0 (
      showName = LFE @position@,
      devName = LFE0,
      posAttrID = SERIAL_NO,
      posAttrValue = 1,
   );
   SLOT PicSlot card_slot0 (
      showName = card slot
@position@,
      devName = card_slot0,
      posAttrID = CARD_ID,
      posAttrValue = 0,
      isContainer = 1,
   );
   PORT EthMGEPort admin_eth_ge0
(
      showName = MEth@position@,
      devName = admin_eth_ge0,
      posAttrID = PORT_ID,
      posAttrValue = 1,
   );
```

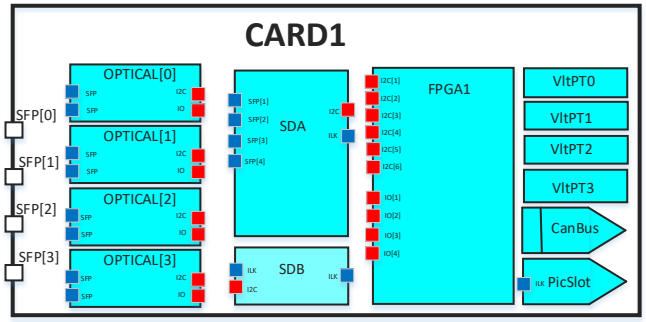
```
SLOT Fe_Slot fe_slot (
      showName = Fe slot @position@,
      devName = fe_slot,
      posAttrID = CARD_ID,
      posAttrValue = 100,
      isContainer = 1,
);
   CHIP ISSU issupreload (
      showName = issupreload,
      devName = issupreload,
      posAttrID = SERIAL_NO,
      posAttrValue = 4294967295,
);
   CHIP EPLD1 epld1ctl (
      showName = epld1ctl @position@,
      devName = epld1ctl,
      posAttrID = SERIAL_NO,
      posAttrValue = 4294967295,
);
    SLOT CPUSLOT CPU_slot[id=1-2] (
        showName = CPU slot @position@,
        devName = CPU_slot@id@,
        posAttrID = CARD_ID,
        posAttrValue = @id@,
        mngMode = REMOTE,
        isContainer = 1,
    );
```

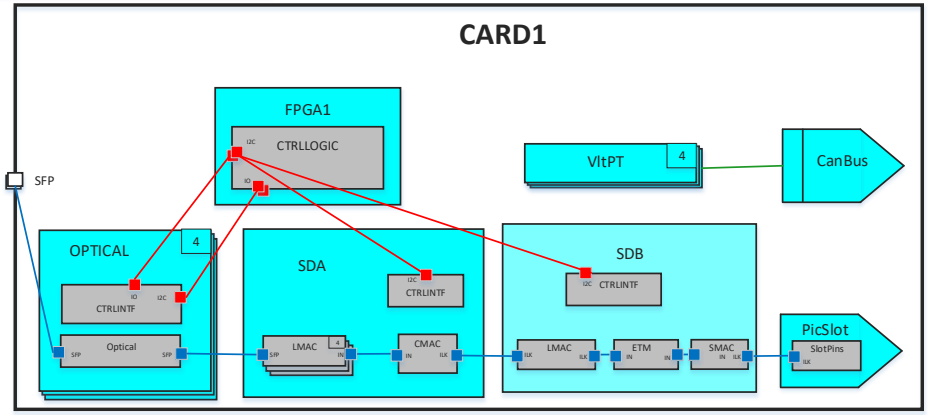
```
CHIP EPLD2 epld2ctl (
      showName = epld1ct2 @position@,
      devName = epld2ctl,
      posAttrID = SERIAL_NO,
      posAttrValue = 4294967295,
  );
CHIP VFMEA fmea_dev (
      showName = MPU slot @position@,
      devName = fmea_dev,
      posAttrID = SERIAL_NO,
      posAttrValue = 4294967295,
  );
 PANELPORT PANELPORT_QSFP28
QSFP28_100gf28[id=0-3] (      showName =
100GEPANELPORT@position@,
 {
 extModule = {qsfp, qsfp28, qsfp28To4xsfpplus,
qsfp28To4xsfp28}
CHIPPORT OPTPIN pin[0-3]   }
PANELPORT PANELPORT_SFP_25GE
SFP28_25gf10[id=0-3] (      showName =
25GEPANELPORT@position@)
{
  extModule  sfp28, rj45;
  CHIPPORT OPTPIN pin[0]   }
CHIP INTERNAL SD5981 sd5981[id=0 - 1]( devName
= sd5981-@id@)
{
    ELEMENT LSW lsw[@id@]()
        {        CHIPPORT UNITPORT unitport[16-
23,32-39]()     }
}
```

```
CHIPLINK DATAPATH links {
QSFP28_100gf28[0].pin[0-3] <--> sd5981[0].lsw.unitport[16-19]
QSFP28_100gf28[1].pin[0-3] <--> sd5981[0].lsw.unitport[20,22,21,23]
QSFP28_100gf28[2].pin[0-3] <--> sd5981[0].lsw.unitport[32-35]
QSFP28_100gf28[3].pin[0-3] <--> sd5981[0].lsw.unitport[36,38,37,39]
SFP28_25gf10[0].pin[0] <--> sd5981[1].lsw.unitport[16]
SFP28_25gf10[1].pin[0] <--> sd5981[1].lsw.unitport[17]
SFP28_25gf10[2].pin[0] <--> sd5981[1].lsw.unitport[18]
 SFP28_25gf10[3].pin[0] <--> sd5981[1].lsw.unitport[19]   }}
CHIP MBUS_COM mbus_com (
   showName = mbus_com @position@,
   devName = mbus_com,
   posAttrID = SERIAL_NO,
   posAttrValue = 4294967295,
);

CHIP VltPT vltPT[id=0-27] (
     showName = sensor @position@,
     devName = vltPT@id@,
     posAttrID = SERIAL_NO,
     posAttrValue = @id@,
  );

 BUS CanBus_t bus71 (phyChannelID=1) {
     mbus_com <--> CHANNEL CanChannel_t ( token0=0, token1=0,
token2=0, token3=1, token4=0, token5=0, token6=0, token7=0)
  }
  TOBUS CanBus_t bus45 (phyChannelID=#_CHASSISID_#) {
     vltPT[0-27] <--> CHANNEL CanChannel_t ( token0=0, token1=0xff,
token2=#_SLOTID_#, token3=#_CHASSISID_#, token4=5, token5=0,
token6=0, token7=[22-49])
  }
}
```

Composition Model

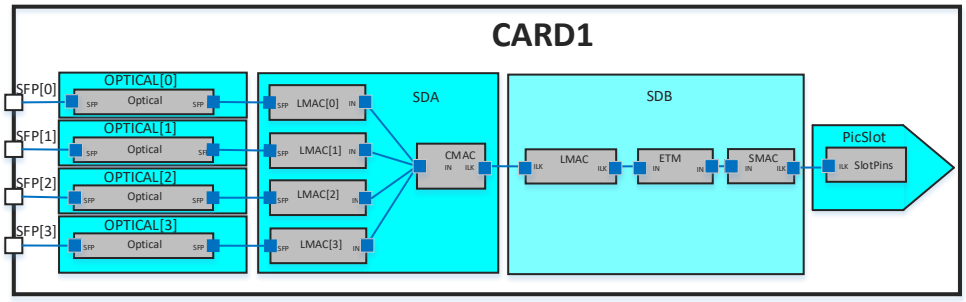
Structured & Composite Diagrams

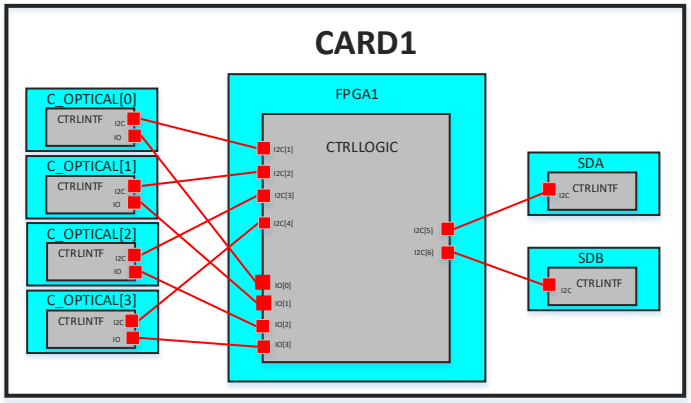




Composition Model

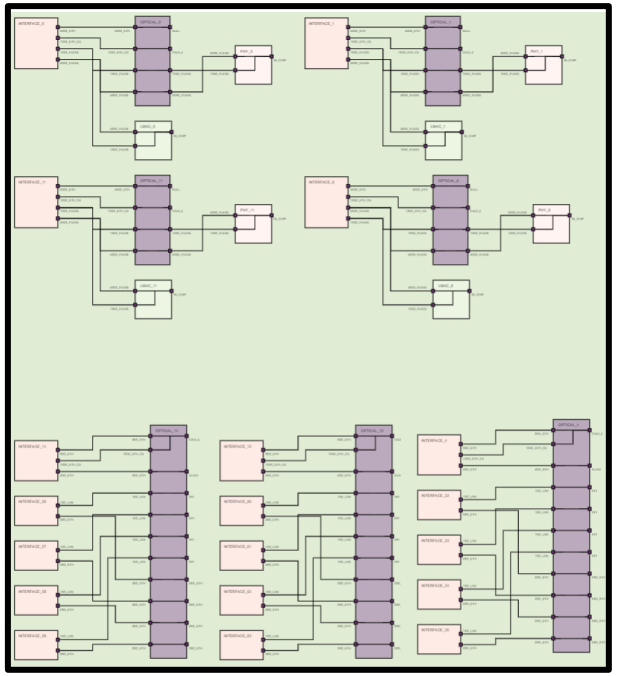
Datapath & Control Diagrams





Optical Model

C-code viewer



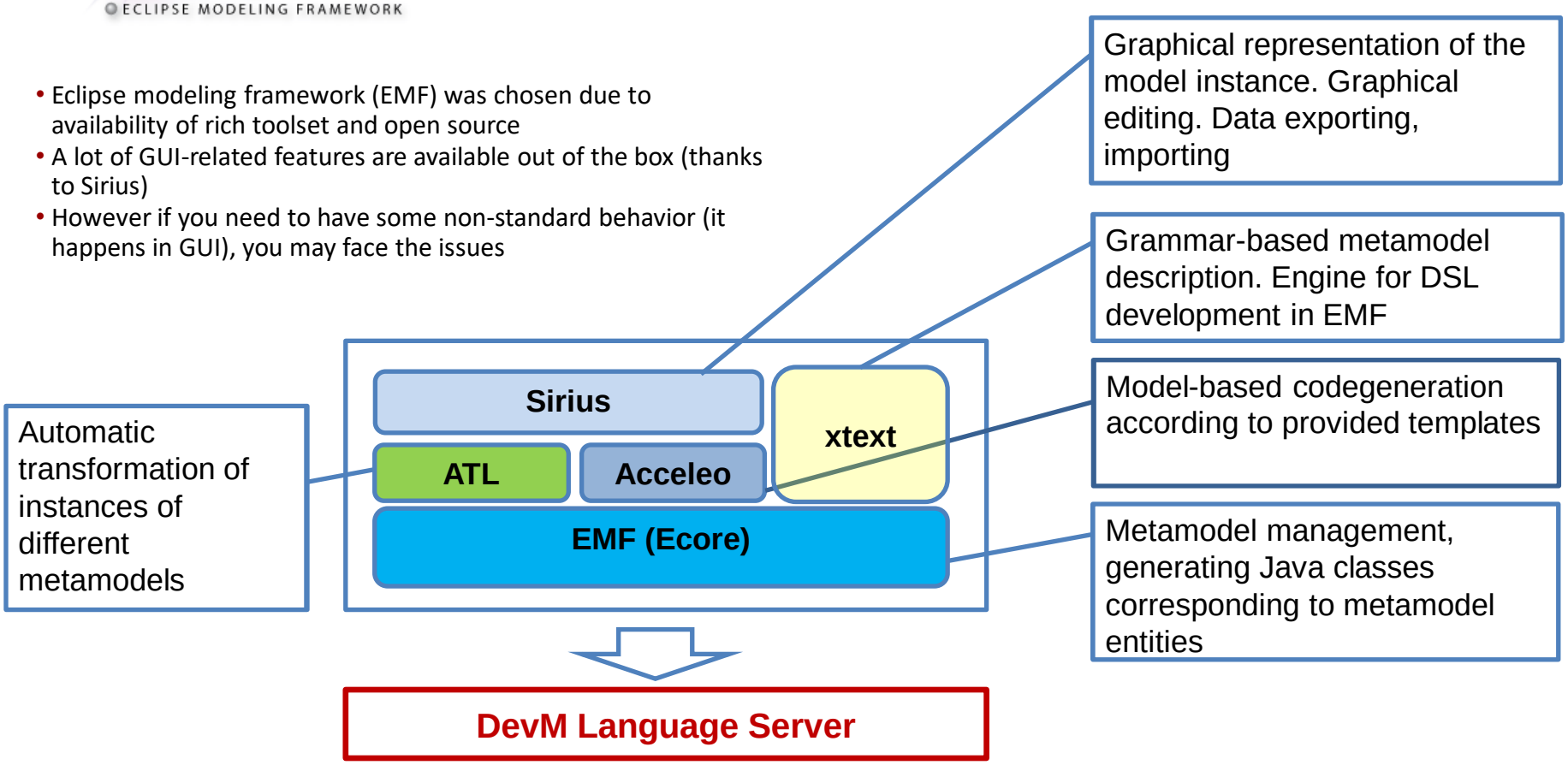
Visualization Tools

- First, diagrams are **automatically generated** by DevM/ specifications or C-programs
- Thus, the visual language is not intended to create specifications, its purpose is **partial visualization** of ones
- Consequently, diagrams are **suitable for discussions, presentations, brushing up**, but not for everyday work
- Finally, visual language could contribute **to decreasing learning curve** during Device Management development process

Root Technologies

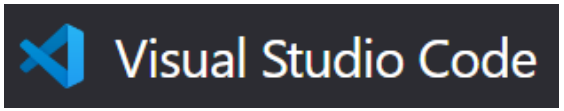
Eclipse Modeling Framework

- Eclipse modeling framework (EMF) was chosen due to availability of rich toolset and open source
- A lot of GUI-related features are available out of the box (thanks to Sirius)
- However if you need to have some non-standard behavior (it happens in GUI), you may face the issues



Root Technologies

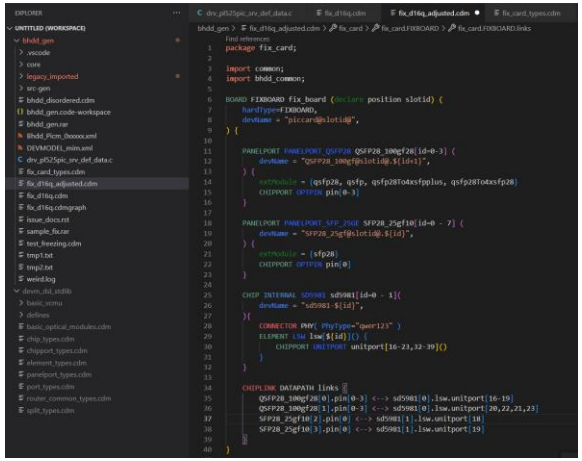
Visual Studio Code



Visual Studio Code

- **DevM Language Server**
- **Spotty, Konva, etc.**
- **C++Language Server**
- **……**

An example of DevM Framework



- We have seamless integration of DevM framework with target product line development environment!

Agenda

- Model-Based Approach
- DSLs
- Product Line approach
- Example
- Conclusions and Discussion

Model-Based DSL

- Model-based approach means designing **strict-defined domain abstractions** for a product line
- Moreover, for this purpose **model-based metaphors** are used
  - Presenting a system as a set of entities and relationships
  - Black-box presentation with external interfaces, ports and channels
  - Types and instances
  - Hierarchy of types and using them
  - Packages (import/export, scopes, conflicts, etc.)
- **Moving far from XML!**
- Thoroughly design abstractions and supporting tools/frameworks are actual for product lines **due to huge resources involved**

Model-Based DSL

- And implement these abstractions in **textual DSL**

And what about visualization?

- First, diagrams are **automatically generated** on textual DSL-programs
- Thus, the visual language is not intended to create specifications, its purpose is **partial visualization** of ones
- Consequently, diagrams are **suitable for discussions, presentations, brushing up**, but not for everyday work
- Finally, visual language could contribute **to decreasing learning curve** during Device Management development process
- Using **ViewToView-transformations**[*)] to create various browsing services

---

**) D. V. Koznov, E. V. Larchik, Andrey N. Terekhov. View to view transformations in domain specific modeling. Program. Comput. Softw. 41(4): 208-214 (2015)*

Conclusion or Lessons to Learn

- Improving existing development process we **can not create absolutely new abstractions**
  - It's necessary to take into account existing system architecture
- **Introducing new tools** into development process is more important than creating academically perfect DSL
  - We need to keep a balance between ideas and practice
- **Elicitation of information** about problem domain takes a lot of efforts
  - Subdomains and architecture of Datacom product lines are really complicated, containing a lot of details and specifics
- New domain-specific tool is developed and introduced in **step-by-step manner** taking a lot of iterations
  - It seems to me we have much more iterations than for a software intended to external customers

Thank you for your attention!

Questions?