Meta-Model and Model co-evolution

Jean-Marie Favre
University of Grenoble
OUTLINE

- Motivation and background: Industry

- Software in 3D
  - D1: meta dimension
  - D2: engineering dimension
  - D3: representation dimension

- Evolution: entering the 4th dimension…

- Conclusion
Part I:

Motivation and Background
Historical mistakes in Software Engineering

- (1) Software is stable
- (2) Software is made of programs

Everything evolve in complex industrial contexts
Architecture and Code co-evolution

- Explicit vs. implicit architecture
- Architecture and code both evolve
- Horizontal impacts
- Vertical impacts
- Synchronization and conformance issues
- Risks of erosion
- Architecture-driven vs. code-driven

- A "well identified" phenomenon nowadays
- Initially neglected by academics
Architecture and Code co-evolution

- Explicit vs. implicit architecture
- Architecture and code both evolve
- Horizontal impacts
- Vertical impacts
- Architecture-driven vs. code-driven
- Synchronization and conformance issues
- Risks of erosion

- A "well identified" phenomenon nowadays
- Initially neglected by academics
Architecture and Code co-evolution

- Explicit vs. implicit architecture
- Architecture and code both evolve
- Horizontal impacts
- Vertical impacts
- Synchronization and conformance issues
- Risks of erosion
- Architecture-driven vs. code-driven

- A "well identified" phenomenon nowadays
- Initially neglected by academics
Program / Language / Tool co-evolution

Program/Compiler co-evolution

Language/Compiler co-evolution

Program/Language co-evolution
Program / Language / Tool co-evolution

Program/Language co-evolution

Language/Compiler co-evolution

Program/Compiler co-evolution
Program / Language / Tool co-evolution
Program / Language / Tool co-evolution
Model / Meta-Model / Tool co-evolution
Schema Evolution

Conceptual schema

data

Implementation schema
Background: A 7-year case study

Collaboration with industry

- World leader in CAD/CAM
- 19,000 clients, 180,000 seats
- Clients: Boeing, Chrysler, …
- Main software: CATIA
CATIA: a very large Software Product Line

- 1200+ software engineers
- 70 000+ classes C++
- 8 000+ components
- 5 000+ interfaces
- 3 000+ DLLs
- 800+ frameworks
- …

Need to raise the levels of abstraction
- Architecture
- Metamodel
## A Meta-Model Driven Architecture Recovery Process

<table>
<thead>
<tr>
<th><strong>Metaware domain and asset analysis</strong></th>
<th>Metaware inventory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Meta-models recovery</td>
</tr>
<tr>
<td></td>
<td>Meta-models Integration</td>
</tr>
<tr>
<td></td>
<td>Meta-model clustering</td>
</tr>
<tr>
<td></td>
<td>Meta-model packaging</td>
</tr>
<tr>
<td><strong>Metaware requirement analysis</strong></td>
<td>Meta-level actors identification</td>
</tr>
<tr>
<td></td>
<td>Meta-level use cases identification</td>
</tr>
<tr>
<td></td>
<td>Metaware assessment</td>
</tr>
<tr>
<td></td>
<td>Metaware Improvement analysis</td>
</tr>
<tr>
<td></td>
<td>Meta-level use cases description</td>
</tr>
<tr>
<td><strong>Metaware specification</strong></td>
<td>Meta-model filtering and extension</td>
</tr>
<tr>
<td></td>
<td>Presentation specification</td>
</tr>
<tr>
<td></td>
<td>Metaware specification packaging</td>
</tr>
<tr>
<td><strong>Metaware implementation</strong></td>
<td>Extractors development and reuse</td>
</tr>
<tr>
<td></td>
<td>Viewers development and reuse</td>
</tr>
<tr>
<td></td>
<td>Extractors and Views integration</td>
</tr>
<tr>
<td><strong>Metaware execution</strong></td>
<td>Execution</td>
</tr>
<tr>
<td><strong>Metaware evolution</strong></td>
<td>Evaluation</td>
</tr>
<tr>
<td></td>
<td>Feedback</td>
</tr>
</tbody>
</table>

---

**www.adele.imag.fr/~jmfavre**

ELISA, September 23, 2003, Amsterdam
Part II:
The 3D Software Space
The 3D Software space

Meta Dimension

Representation Dimension

Engineering Product Dimension
The 3D Software space

- D1: Meta
- MMM: Meta-Meta-Model
- MM: Meta-Model
- M: Model
- I: Instance

Engineering Product Dimension
The 3D Software space
The 3D Software space

**D3** Representation

- **IR** Implicit repr.
- **AR** Abstract repr.
- **CR** Concrete repr.

**Meta Dime**

**Engineering Product Dimension**
A taxonomy of software artefacts
D1: The Meta dimension

- The Meta-towers
- The Meta-pyramid
- The Meta actor pyramid
D1: The Meta-towers

- Meta-grammar
  - e.g. BNF
- Grammar
  - e.g. Java
- Sentence
  - e.g. Java program

- Meta-model
  - e.g. UML meta-model
- Model
  - e.g. a class diagram of the ACME Banking app.
- Instance
  - e.g. the state of the bank on Tue 24, Dec, 2pm

- Data model
  - e.g. Relational data model
- Schema
  - e.g. the schema of the ACME Banking database
- Data
  - e.g. the data of the ACME banks on Tue 24, Dec, 2pm

Languages
UML/MDA
Databases
D1: The Meta-pyramid
D1: The Meta-pyramid
D1: The Meta-pyramid

METAWARE

- Meta-Meta-Model
- Meta-Model
- Model
- Instance

Alternatives
- ACME Banking Apps
- FOO Banking Apps
- FOO Banking State 10am
- ACME Banking State 3pm
- Z Bookstore Apps
- Z Bookstore State 9pm

Real world

# of entities
D1: The Meta-pyramid
D1: The Meta actor pyramid
D2: The Engineering Dimension

- The Engineering-tower
- The Engineering-pyramid
- The Engineering actor pyramid

Engineering Product Dimension
D2: The Engineering Pyramid
D1+D2: Meta + Engineering

D3

Appliware execution

Software

Appliware

Metaware

D2

R

Tue 24 Dec, 10pm
Tom wants to withdraw 100€ from the cash machine located "12 rue de la monnaie" at Bruxelles.

A

The XB12 feature is running on the cash machine. The cash machine is connected to the ACME bank server executable XP23Serv via a secure connector.

D

class client implements Serializable {
    private String name;
    private Vector Accounts;
    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        if (name == null)
            throw new NullPointerException();
        this.name = name;
    }

I

Instances

Models

Meta-models

Meta-meta Models

References to objects

Java objects

D1+D2: Meta + Engineering

www-adele.imag.fr/~jmfavre

ELISA, September 23, 2003, Amsterdam
The XB12 feature is running on the cash machine. The cash machine is connected to the ACME bank server executable XP23Serv via a secure connector.

Tue 24 Dec, 16pm
Tom wants to withdraw 100€ from the cash machine located "12 rue de la monnaie" at Bruxelles

Client: Tom
- name: Tom
- account balance: 24800
- account balance: 300

public class Client {
  private String name;
  private Vector Accounts;
  public String getName() {
    return this.name;
  }
  public void setName(String name) {
    if (name == null) throws new NullPointerException();
    this.name = name;
  }
  ...
}
Tue 24 Dec, 10pm
Tom wants to withdraw 1006 from the cash machine located "12 rue de la monnaie" at Bruxelles.

The XB12 feature is running on the cash machine. The cash machine is connected to the ACME bank server executable XP23Serv via a secure connector.

tom : Client
name = "tom"
balance = 24600

g231 : Account
balance = 24600

g2204 : Account
balance = 300

class Client implements Serializable {
    private String name;
    private Vector Accounts;
    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        if (name == null)
            throw new NullPointerException();
        this.name = name
    }
    ...
}
Tom wants to withdraw 100€ from the cash machine located "12 rue de la monnaie" at Bruxelles.

R1: All transfers must be secured.
R2: Clients can transfer money either via cash machines and internet.
R3: Withdraw requires previous identification.

The X612 feature is running on a cash machine to validate the AC and execute a secure withdrawal.

public void setName(String name) {
    if (name == null) {
        throw new NullPointerException();
    }
    this.name = name;
}
Tue 24 Dec, 10pm
Tom wants to withdraw
100€ from the cash machine
located "12 rue de la
monnaie" at Bruxelles.

The XB12 feature is running
on the cash machine. The cash
machine is connected to
the ACME bank server
executable XP23Serv via
a secure connector.
Tom wants to withdraw 100€ from the cash machine located "12 rue de la monnaie" at Bruxelles.

The XB12 feature is running on the cash machine. The cash machine is connected to the ACME bank server executable XP23Serv via a secure connection.

class Client implements Serializable {
    private String name;
    private Vector accounts;
    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        if (name == null)
            throw new NullPointerException();
        this.name = name;
    }...

    Client() {
        accounts = new Vector();
    }...
}
Tue 24 Dec, 10pm
Tom wants to withdraw 100€ from the cash machine located "12 rue de la monnaie" at Bruxelles.

The XB12 feature is running on the cash machine. The cash machine is connected to the ACME bank server executable xp23serv via a secure connector.

does client implements Serializable {
    private String name;
    private Vector Accounts;
    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        if (name == null)
            throw new NullPointerException();
        this.name = name;
    }
    ...
Tue 24 Dec, 10pm
Tom wants to withdraw 100€ from the cash machine located "12 rue de la monnaie" at Bruxelles.

The XB12 feature is running on the cash machine. The cash machine is connected to the ACME bank server executable XP233serv via a secure connector.

R1: all transfers must be secured
R2: clients can transfer money either via cash machines and internet
R3: withdraw requires previous identification
Tue 24 Dec, 10pm
Tom wants to withdraw 100€ from the cash machine located "12 rue de la monnaie" at Brussels.

The XB12 feature is running on the cash machine. The cash machine is connected to the ACME bank server executable XP23Serv via a secure connector.

R1: all transfers must be secured
R2: clients can transfer money either via cash machines and internet
R3: withdrew requires previous identification
The XB12 is a middle-tier running on the client machine. The ACME executable XObjects use a secure connection.

```java
class Client {
    private String name;
    private Vector accounts;
    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        if (name == null)
            throw new NullPointerException();
        this.name = name;
    }
}
```
Tue 24 Dec, 10pm
Tom wants to withdraw
100€ from the cash machine
located "12 rue de la monnaie" at Bruxelles.

The XB12 feature is running on the cash
machine. The cash
machine is connected to
the ACME bank server
executable XP\$2Serv via
a secure connector.

class client implements Serializable {
    private String name;
    private Vector accounts;
    public String getName() {
        return this.name;
    }

    public void setName( String name ) {
        if (name == null)
            throws new NullPointerException();
    }

    this.name = name

...
Tue 24 Dec, 10pm
Tom wants to withdraw 100€ from the cash machine located "12 rue de la monnaie" at Bruxelles.

The XB12 feature is running on the cash machine. The cash machine is connected to the ACME bank server executable XP23Serv via a secure connector.

class Client implements Serializable {
    private String name;
    private Vector accounts;
    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        if (name == null) throws new NullPointerException();
        this.name = name;
    }
    ...
}
Tuesday Dec 24, 10pm
Tom wants to withdraw 100€ from the cash machine located "12 rue de la manse" at Bruxelles.

The XR12 feature is running on the cash machine. The cash machine is connected to the ACME bank server executable XP/32Serv via a secure connector.

Tom: Client
name = "Tom"

S221: Account
balance = 24000

S2204: Account
balance = 300

class Client implements Serializable {
    private String name;
    private Vector Accounts;
    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        if (name == null)
            throws new NullPointerException();
        this.name = name;
    }
    *

ELISA, September 23, 2003, Amsterdam
Tue 24 Dec, 10pm
Tom wants to withdraw
100€ from the cash machine
located "12 rue de la monnaie" at Bruxelles.

The XB12 feature is running on the cash
machine. The cash machine is connected to
the ACME bank server executable XP23Serv via
a secure connector.

```
client:
    withdraw
    TransferMoney

Cash Machine
<<topic>>

Bank Server
```

```
Client
    1
    client : accounts

Account
    balance : int
```

```
class client implements Serializable {
    private String name ;
    private Vector accounts ;
    public String getName() {
        return this.name ;
    }
    public void setName( String name ) {
        if (name == null)
            throws new NullPointerException() ;
        this.name = name
        ...
```
D3: The Representation Dimension

- The Representation Towers
- The Representation Pyramid
- The Representation actors

Meta Dimension

Representation Dimension

Engineering Product Dimension

- D3
- D2
D3: The Representation Towers

Conceptual Model
 Specification Model
 Implementation Model

Language
 Abstract grammar
 Concrete grammar

Conceptual Schema
 Logical Schema
 Physical Schema

UML – Fowler
 Languages
 Databases
D3: The Representation Dimension
www-adele.imag.fr/~jmfavre

ELISA, September 23, 2003, Amsterdam
8.1.4 Superinterfaces

The optional implements clause in a class declaration lists the names of interfaces that are direct superinterfaces of the class being declared.

Each InterfaceType must name an accessible (§6.6) interface type, or a compile-time error occurs.

A compile-time error occurs if the same interface is mentioned two or more times in a single implements clause.

This is true even if the interface is named in different ways; for example, the
8.1.4 Superinterfaces

The optional implements clause in a class declaration lists the names of interfaces that are direct superinterfaces of the class being declared:

Each InterfaceType must name an accessible (§6.6) interface type, or a compile-time error occurs.

A compile-time error occurs if the same interface is mentioned two or more times in a single implements clause.

This is true even if the interface is named in different ways; for example, the
context Class
   inv i3 : isClass xor isInterface
   inv i4 : isInterface implies superclass->isEMPTY
   inv i5 : superclass->notEmpty implies superclass->isClass
   inv i6 : c.interfaces->forall( i | i.isInterface )

Class
   name : String
   isClass : boolean
   isInterface : boolean

Package
   name : String

Implementation meta-model
   post : result = isClass
   post : result = not isClass
   post : result = super

Specification meta-model
   post : result = toString().concat(package.name)
class Class {
    private String shortName;
    private boolean isClass;
    private Class superClass;
    private Vector retrieveInterfaces();
    public String isClass() { return isClass; }
    public boolean isInterface() { return !isClass; }
    public Class getSuperclass() { return superClass; }
    public String getName() { return shortName + getPackage().getName(); }
    ...
    Class() { .. }
}

post : result = shortName + getPackage().getName()
Appliware execution

Tom wants to withdraw 100€ from the cash machine located "12 rue de la monnaie" al Bruxelles.

The XB12 feature is running on the cash machine. The cash machine is connected to the ACME bank server executable XP23Sen via a secure connector.

D3

Software

Appliware

Metaware

D2

R

tom : Client
name = "tom"

a231 : Account
balance = 24600

a2204 : Account
balance = 300

D

A

Client

1 * account

Account

balance : int

<topic>

Bank Server

Cash Machine

actor

withdraw

transferMoney

client

R1: all transfers must succeed
R2: clients can transfer money either via cash machines and internet
R3: withdraw requires previous identification

act

feature

host

executable

connector

component

subsystem

class

association

associationEnd

javaClass

javaField

javaMethod

statement

expression

package

boxes

lines

symbol

ELISA, September 23, 2003, Amsterdam
Classifying Software Artefacts

- Using the 3D Framework to classify software artefacts
- Coordinates in the reverse order D3-D2-D1

Examples:
- CR-A-MM
- AR-D-M
- …
Classifying Software Transformations

- Transformations or processes as paths in 3D
- Useful to classify SE tools and methods

- Forward engineering
- Reverse engineering
- Evolution & co-evolution
Evolution:

Entering the fourth dimension...
Evolution: Entering the Fourth Dimension (D4)
Co-evolution along the engineering dimension
D2+D4
Co-evolution along the meta dimension
D1+D4
Meta-model / model co-evolution at DS

- Incremental definition of a proprietary component technology
- Incremental implementation of tools by the tool support team
- Production of component-based software at the same time

- Meta-models should be versionned
- Different variants of the meta-model used
  - by different teams within DS
  - by partner companies
- Co-evolution managed in ad-hoc way
- Manual or semi-automatic transformation
Conclusion
Conclusion

- Many academic issues related to meta-modeling / meta-programming
- More issues coming from industry
- Co-evolution of meta-models and models
- Reverse engineering meta-models
- Tool support is required

Supporting evolution at various level is an important requirement for the success of model-driven approaches (e.g. MDA)

Meta-model for evolution vs. evolution of meta-models
Call For Papers

First International Workshop on Meta-models and Schemas for Reverse Engineering

November 13, 2003, Victoria, BC, Canada
www-adele.imag.fr/atem2003

With WCRE'2003

Organizers
Jean-Marie Favre, University of Grenoble, France
Mike Godfrey, University of Waterloo, Canada
Andreas Winter, University Koblenz-Landau, Germany