Data Validation Constraints in MDE

Alessandro Rossini[1], Adrian Rutle[2], Federico Mancini[1], Dag Hovland[1], Khalid A. Mughal[1], Yngve Lamo[2], Uwe Wolter[1]

[1]Department of Informatics, University of Bergen, Norway

[2]Faculty of Engineering, Bergen University College, Norway

16 October 2009
NWPT 2009, Lyngby, Denmark

HØGSKOLEN I BERGEN

Outline

Software Security

- Society relies on software systems
- Violating systems threaten economy, politics and health
- Software security:
  - confidentiality, integrity and availability of systems

Software Security in Software Development

- Present-day software development
  - security neglected: budget constraints and lack of skills
  - security concerns considered too late
- Ideal software development
  - security from the early phases of the development
  - potential security flaws revealed before a software system is implemented

Model-Driven Engineering (MDE)

- Software models as abstract representations of software systems
  - reasoning at a high level of abstraction
- Model-Driven Engineering (MDE)
  - models as primary artefacts of the software development process
  - generation of systems by means of model transformations

MDE Standards: State-of-the-art

- Modelling languages
  - Unified Modeling Language (UML)
  - Eclipse Modeling Framework (EMF)

  Usually graph-based languages
- Constraint languages
  - Object Constraint Language (OCL)

  Usually text-based languages

Challenge

- Specification of security constraints within software models
- Constraints for data validation:
  - process of ensuring that a system operates on correct and meaningful data
  - most common web application security weakness

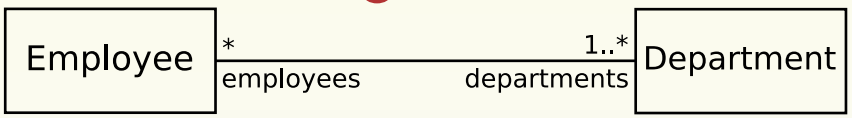
Our contribution

- Specification of data validation constraints on multiple, interdependent structural features of models
  - Diagram Predicate Framework (DPF) as the formal underpinning of our approach to modelling
- Data validation constraints at the model level mapped to corresponding tests at the code level
  - SHIP Validator as a Java based framework for the validation of properties of Java objects

Outline

Project management example

UML class diagram

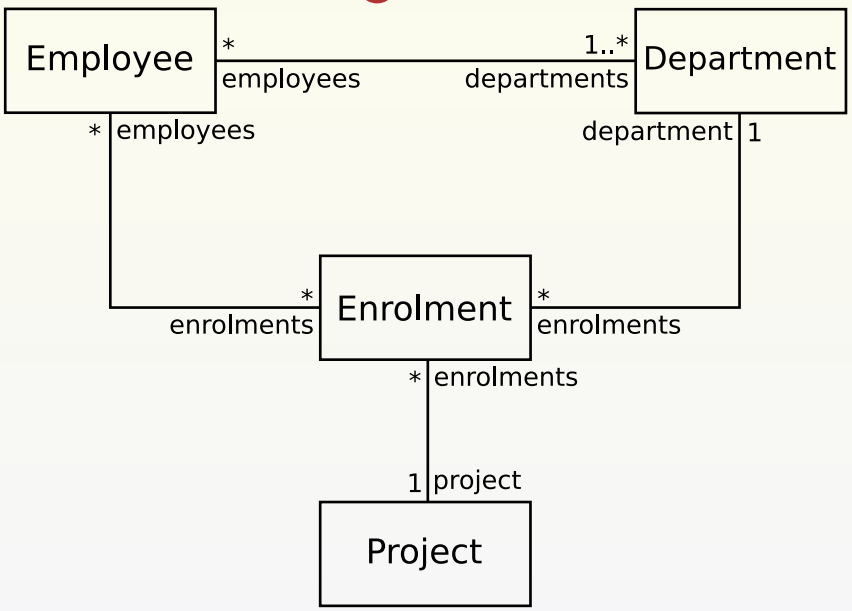


Requirements

1. An employee must work for at least one department.
2. A department may have none or many employees.

Project management example

UML class diagram



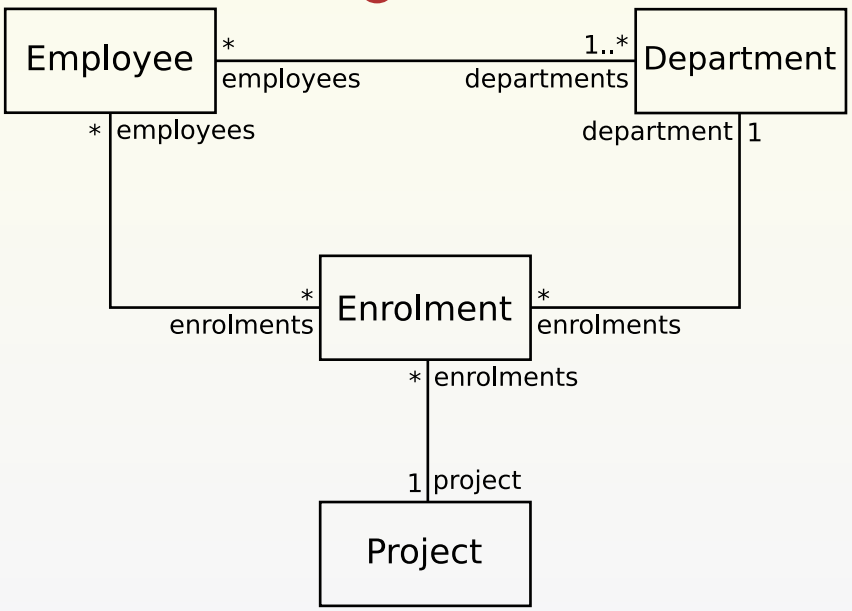
Requirements

3 An employee may be enrolled in none or many projects.

4 A department may control none or many projects.

Project management example

UML class diagram



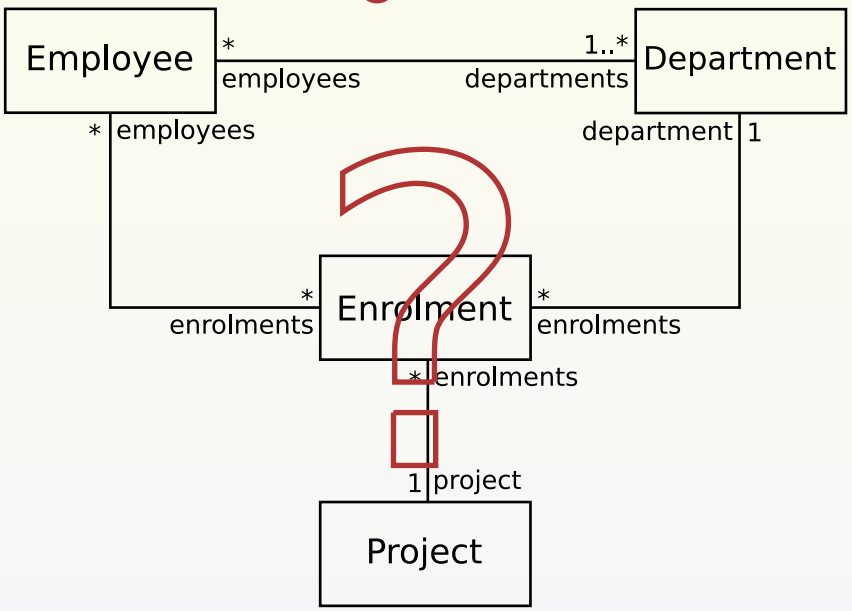
Requirements

5 An employee enrolled in a project must work in the controlling department.

6 A set of employees working for a controlling department must not be enrolled in the same controlled project more than once.

Project management example

UML class diagram



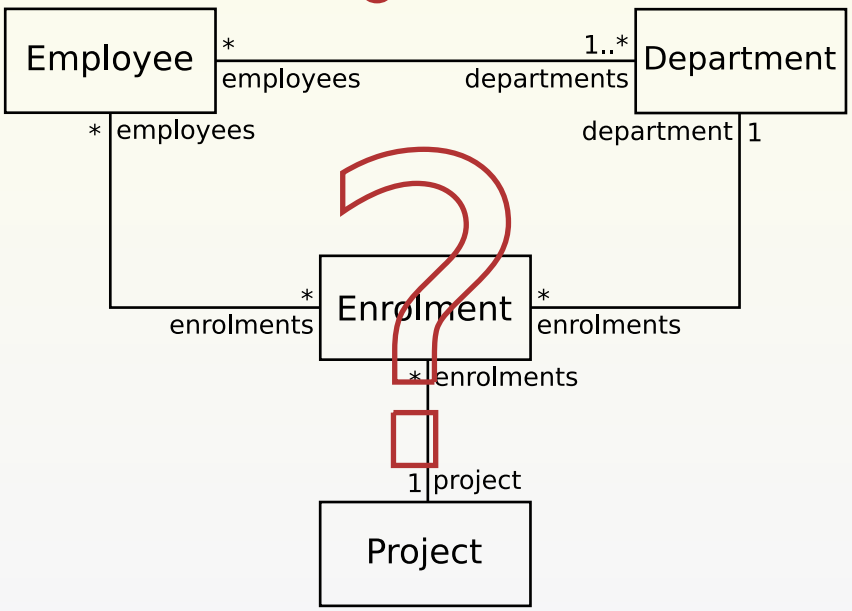
Requirements

5 An employee enrolled in a project must work in the controlling department.

6 A set of employees working for a controlling department must not be enrolled in the same controlled project more than once.

Project management example

UML class diagram



OCL

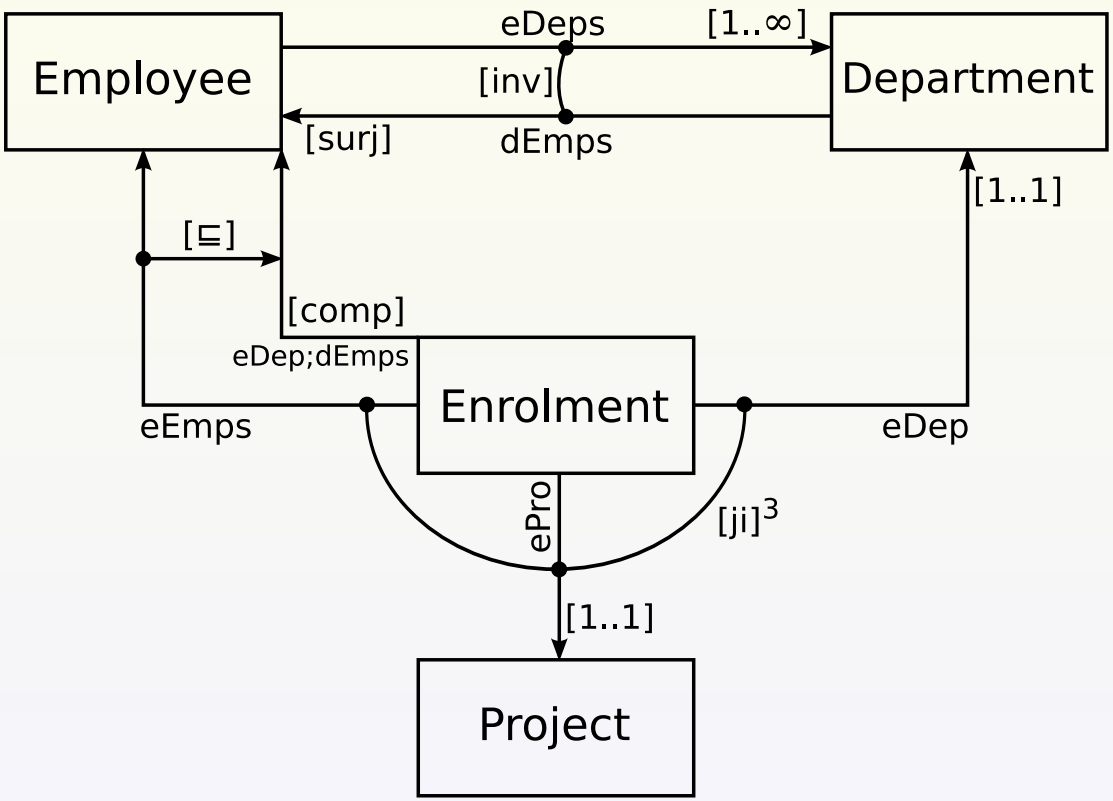
```
context Enrolment
 inv rule6: self.department.employees ->
  includesAll(self.employee)
 inv rule7: Let enrolments:Set(Enrolment)=
   Enrolment.allInstances in
   (not enrolment->exists(enr|
    enr.project=self.project
    and enr.department=self.department
    and enr.employees=self.employees))
```

Requirements

5 An employee enrolled in a project must work in the controlling department.

6 A set of employees working for a controlling department must not be enrolled in the same controlled project more than once.

Proposed solution: constraints integrated

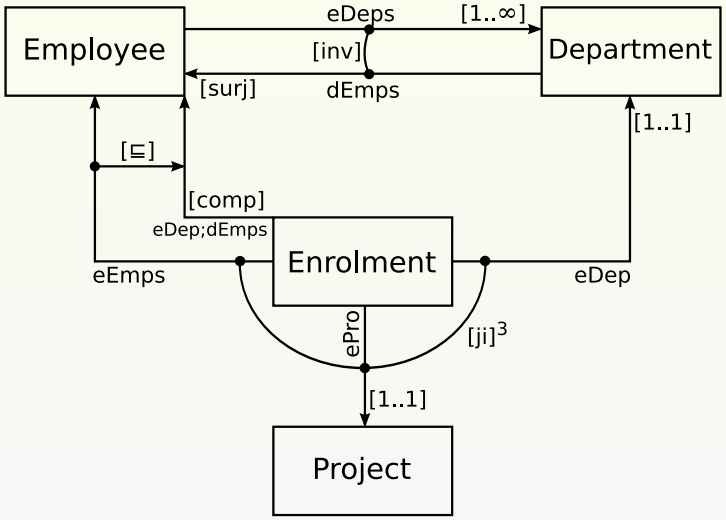


Formalisation approach

- Diagram Predicate Framework (DPF)
- Based on category theory and generalized sketches formalism
  - models: graphs
  - constraints: user-defined diagrammatic predicate signatures

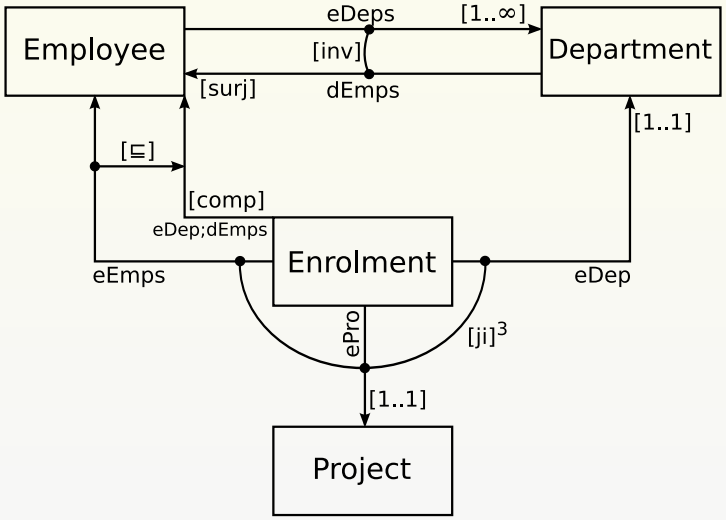
Σ_{struct}-specification

$$S = (G^S, C^S)$$

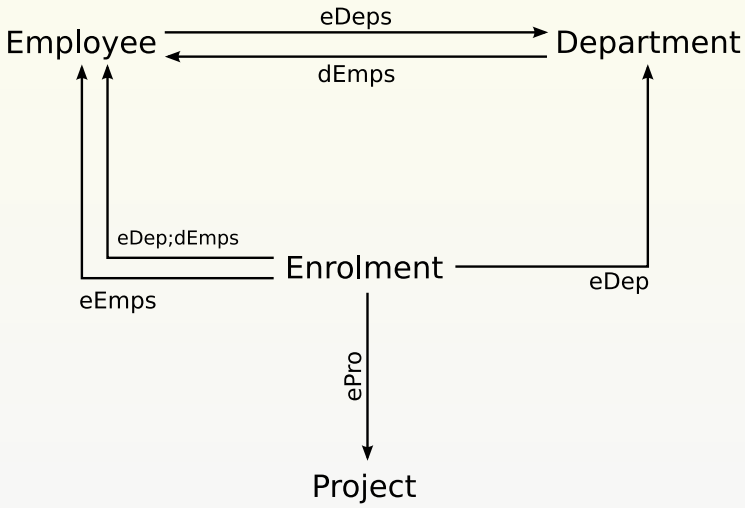


Σ_{struct}-specification

$S = (G^S, C^S)$

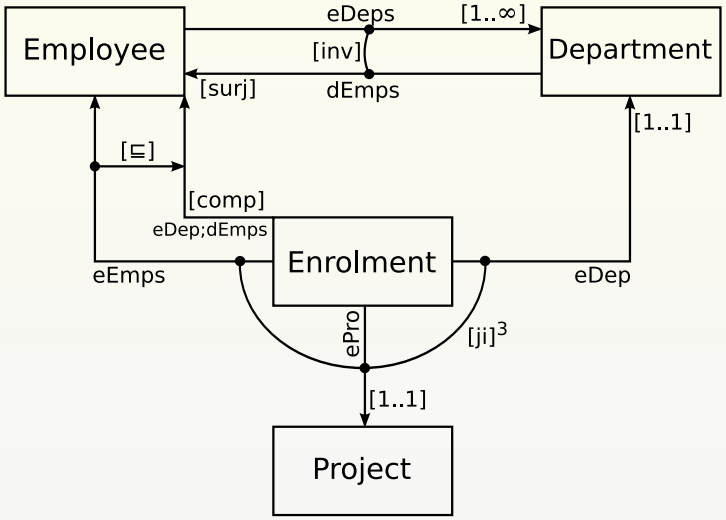


G^S

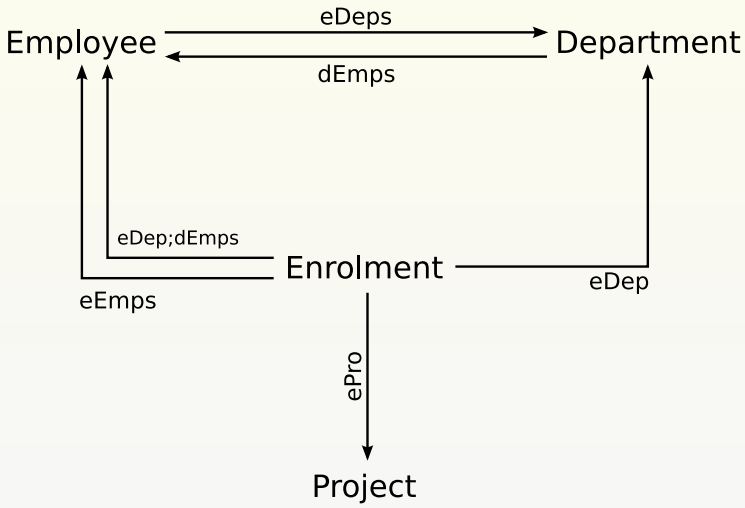


Σ_{struct}-specification

$S = (G^S, C^S)$



G^S

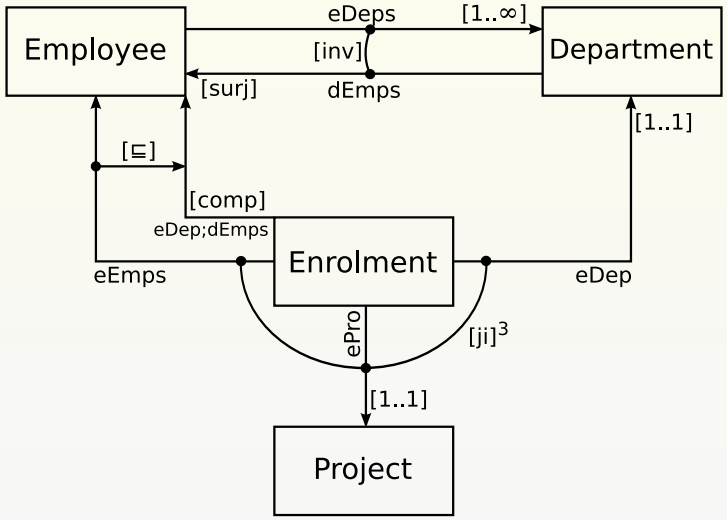


$\Sigma_{struct} = (\Pi, \alpha)$

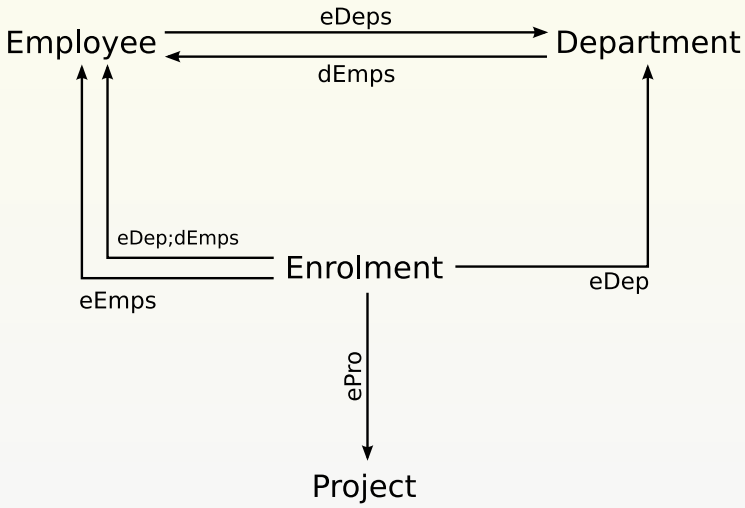
Π	α	Proposed visual.	Intended semantics
`[mult(n,m)]`	$1 \xrightarrow{x} 2$	$\boxed{X} \xrightarrow[\text{[n..m]}]{f} \boxed{Y}$	$\forall x \in X : n \leq \lvert f(x) \rvert \leq m$
`[inverse]`	$1 \underset{y}{\overset{x}{\rightleftarrows}} 2$	$\boxed{X} \underset{g}{\overset{f}{\rightleftarrows}} \boxed{Y}$ [inv]	$\forall x \in X$, $\forall y \in Y : y \in f(x)$ iff $x \in g(y)$

Σ_{struct}-specification

$S = (G^S, C^S)$



G^S



C^S

(p, δ)	$\alpha(p)$	$\delta(\alpha(p))$
([mult(1,∞)], δ_1)	$1 \xrightarrow{x} 2$	$Employee \xrightarrow{eDeps} Department$
([inverse], δ_2)	$1 \underset{y}{\overset{x}{\rightleftarrows}} 2$	$Employee \underset{dEmps}{\overset{eDeps}{\rightleftarrows}} Department$

$\Sigma_{struct} = (\Pi, \alpha)$

Π	α	Proposed visual.	Intended semantics
[mult(n,m)]	$1 \xrightarrow{x} 2$	$\boxed{X} \xrightarrow[{[n..m]}]{f} \boxed{Y}$	$\forall x \in X : n \leq \lvert f(x) \rvert \leq m$
[inverse]	$1 \underset{y}{\overset{x}{\rightleftarrows}} 2$	$\boxed{X} \underset{g}{\overset{f}{\underset{[inv]}{\rightleftarrows}}} \boxed{Y}$	$\forall x \in X$, $\forall y \in Y : y \in f(x)$ iff $x \in g(y)$

Signature Σ_{struct}

Π	α	Proposed visualisation	Intended semantics
`[mult(n,m)]`	$1 \xrightarrow{x} 2$	$\boxed{X} \xrightarrow[\text{[n..m]}]{f} \boxed{Y}$	$\forall x \in X : n \leq \lvert f(x) \rvert \leq m$
`[injective]`	$1 \xrightarrow{x} 2$	$\boxed{X} \xrightarrow[\text{[inj]}]{f} \boxed{Y}$	$\forall x, x' \in X : f(x) = f(x')$ implies $x = x'$
`[surjective]`	$1 \xrightarrow{x} 2$	$\boxed{X} \xrightarrow[\text{[surj]}]{f} \boxed{Y}$	$f(X) = Y$
`[jointly-injective]`^{3}	$\begin{tikzcd} 1 \arrow[r, "x"] \arrow[d, "z"'] \arrow[dr, "y"] & 2 \\ 4 & 3 \end{tikzcd}$	$\begin{tikzcd} \boxed{X} \arrow[r, "f"] \arrow[d, "h"'] \arrow[dr, "g"] \arrow[dr, phantom, "\text{[ji]}^3"] & \boxed{Y} \\ \boxed{Æ} & \boxed{Z} \end{tikzcd}$	$\forall x, x' \in X : f(x) = f(x')$ and $g(x) = g(x')$ and $h(x) = h(x')$ implies $x = x'$
`[inverse]`	$\begin{tikzcd} 1 \arrow[r, bend left, "x"] & 2 \arrow[l, bend left, "y"] \end{tikzcd}$	$\begin{tikzcd} \boxed{X} \arrow[r, bend left, "f"] \arrow[r, phantom, "\text{[inv]}"] & \boxed{Y} \arrow[l, bend left, "g"] \end{tikzcd}$	$\forall x \in X$, $\forall y \in Y : y \in f(x)$ iff $x \in g(y)$
`[subset]`	$\begin{tikzcd} 1 \arrow[r, bend left, "x"] \arrow[r, bend right, "y"'] & 2 \end{tikzcd}$	$\begin{tikzcd} \boxed{X} \arrow[r, bend left, "f"] \arrow[r, phantom, "[\sqsubseteq]"] \arrow[r, bend right, "g"'] & \boxed{Y} \end{tikzcd}$	$\forall x \in X : f(x) \subseteq g(x)$

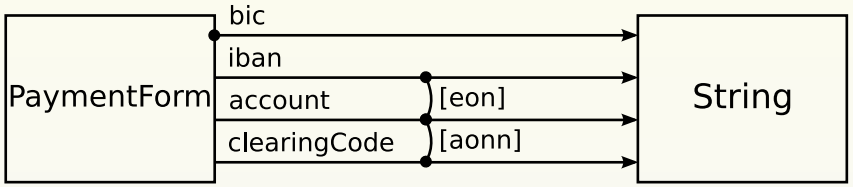
Outline

International money transfers

- *IBAN*: standard for identifying bank accounts internationally
- Countries not adopting IBAN: *clearing code* together with *account number*
- *BIC*: standard for identifying banks globally

Payment form example

Diagrammatic specification

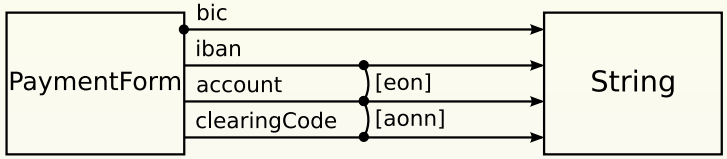


Requirements

1. The BIC code of the beneficiary's bank is required
2. Either the IBAN or both clearing code and the account number are required

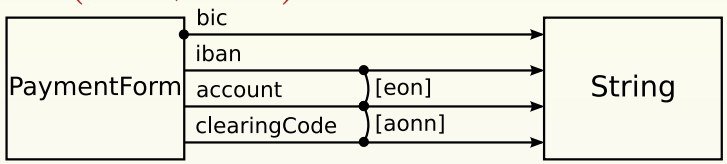
Σ_{sec}-specification

$$S = (G^S, C^S)$$

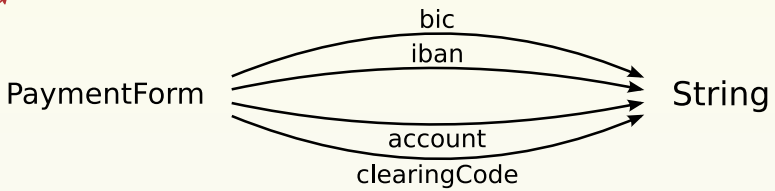


Σ_{sec}-specification

$S = (G^S, C^S)$

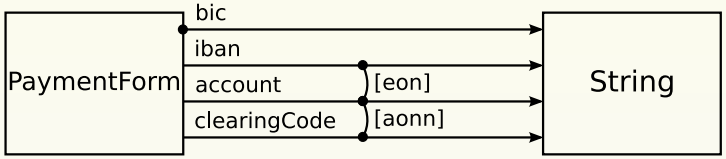


G^S

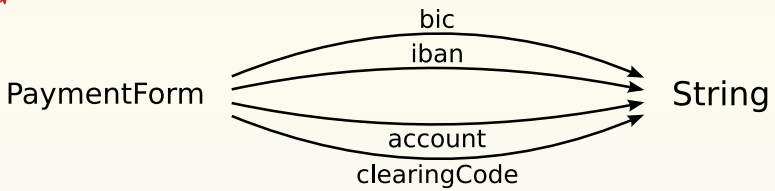


Σ_{sec}-specification

$$S = (G^S, C^S)$$



G^S

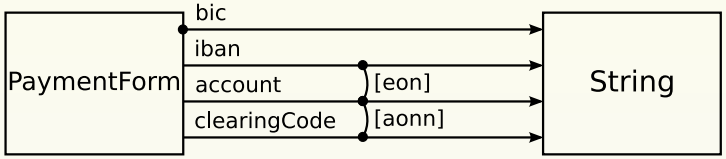


$$\Sigma_{sec} = (\Pi, \alpha)$$

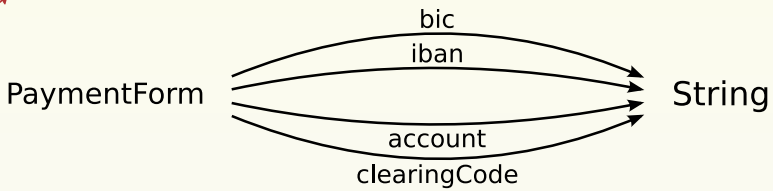
Π	α	Proposed visual.	Intended semantics
[required]	$1 \xrightarrow{x} 2$	$\boxed{X} \xrightarrow{f} \boxed{Y}$	$\forall x \in X : \lvert f(x) \rvert \geq 1$
[exactly-one-null]^2	$1 \xrightarrow{x} 2$, $1 \xrightarrow{y} 3$	$\boxed{X} \xrightarrow{f} \boxed{Y}$, $\boxed{X} \xrightarrow{g} \boxed{Z}$ [eon]	$\forall x \in X : (\lvert f(x) \rvert \geq 1$ and $\lvert g(x) \rvert = 0)$ or $(\lvert f(x) \rvert = 0$ and $\lvert g(x) \rvert \geq 1)$

Σ_{sec}-specification

$S = (G^S, C^S)$



G^S



C^S

(p, δ)	$\alpha(p)$	$\delta(\alpha(p))$
([required], δ_1)	$1 \xrightarrow{x} 2$	$PaymentForm \xrightarrow{bic} String$
([exactly-one-null], δ_2)	$1 \overset{x}{\underset{y}{\rightrightarrows}} 2$	$PaymentForm \overset{account}{\underset{clearingCode}{\rightrightarrows}} String$

$\Sigma_{sec} = (\Pi, \alpha)$

Π	α	Proposed visual.	Intended semantics
[required]	$1 \xrightarrow{x} 2$	$X \xrightarrow{f} Y$	$\forall x \in X : \lvert f(x) \rvert \geq 1$
[exactly-one-null]^2	$1 \xrightarrow{x} 2$, $1 \xrightarrow{y} 3$	$X \xrightarrow{f} Y$, $X \xrightarrow{g} Z$ [eon]	$\forall x \in X : (\lvert f(x) \rvert \geq 1$ and $\lvert g(x) \rvert = 0)$ or $(\lvert f(x) \rvert = 0$ and $\lvert g(x) \rvert \geq 1)$

Signature Σ_{sec}

Π	α	Proposed visualisation	Intended semantics
`[required]`	$1 \xrightarrow{x} 2$	$\boxed{X} \overset{f}{\bullet\!\!\longrightarrow} \boxed{Y}$	$\forall x \in X : \lvert f(x) \rvert \geq 1$
`[exactly-one-null]`^{2}	$\begin{tikzcd} 1 \arrow[r, "x"] \arrow[d, "y"] & 2 \\ 3 & \end{tikzcd}$	$\begin{tikzcd} \boxed{X} \arrow[r, "f"] \arrow[d, "g"'] & \boxed{Y} \\ \boxed{Z} \arrow[ur, bend right, "\text{[eon]}"'] & \end{tikzcd}$	$\forall x \in X : (\lvert f(x) \rvert \geq 1$ and $\lvert g(x) \rvert = 0)$ or $(\lvert f(x) \rvert = 0$ and $\lvert g(x) \rvert \geq 1)$
`[all-or-none-null]`^{2}	$\begin{tikzcd} 1 \arrow[r, "x"] \arrow[d, "y"] & 2 \\ 3 & \end{tikzcd}$	$\begin{tikzcd} \boxed{X} \arrow[r, "f"] \arrow[d, "g"'] & \boxed{Y} \\ \boxed{Z} \arrow[ur, bend right, "\text{[aonn]}"'] & \end{tikzcd}$	$\forall x \in X : (\lvert f(x) \rvert \geq 1$ and $\lvert g(x) \rvert \geq 1)$ or $(\lvert f(x) \rvert = 0$ and $\lvert g(x) \rvert = 0)$

Java class with annotation

```
public class PaymentForm {
  String bic, iban, account, clearingCode;

  @Required
  public String getBic() { return bic; }

  @ExactlyOneNull
  @NotRequired
  public String getIban() { return iban; }

  @ExactlyOneNull
  @AllOrNoneNull
  @NotRequired
  public String getAccount() { return account; }

  @AllOrNoneNull
  @NotRequired
  public String getClearingCode() { return clearingCode; }
}
```

- SHIP Validator tests represented by annotations
  - e.g. `@Required`: input from the user must be available via getter methods
- Relevant getter methods annotated
- Corresponding tests run when the object is passed to the Validator at runtime

Outline

Related work

- *Jürjens*
  - Secure Systems Development with UML
  - Usage of UML extension mechanisms

Summary

- DPF enables constraining multiple, interdependent structural features of models
  - no need for attached OCL constraints
- Data validation constraints at the model level mapped to corresponding tests at the code level

Future work

- Logic for dependencies between predicates
- Prototype tool
- Fully-fledged case study

Thank you!

Questions?