Contextual constraints in configuration languages

Dennis Wagelaar
System and Software Engineering Lab

Vrije Universiteit Brussel

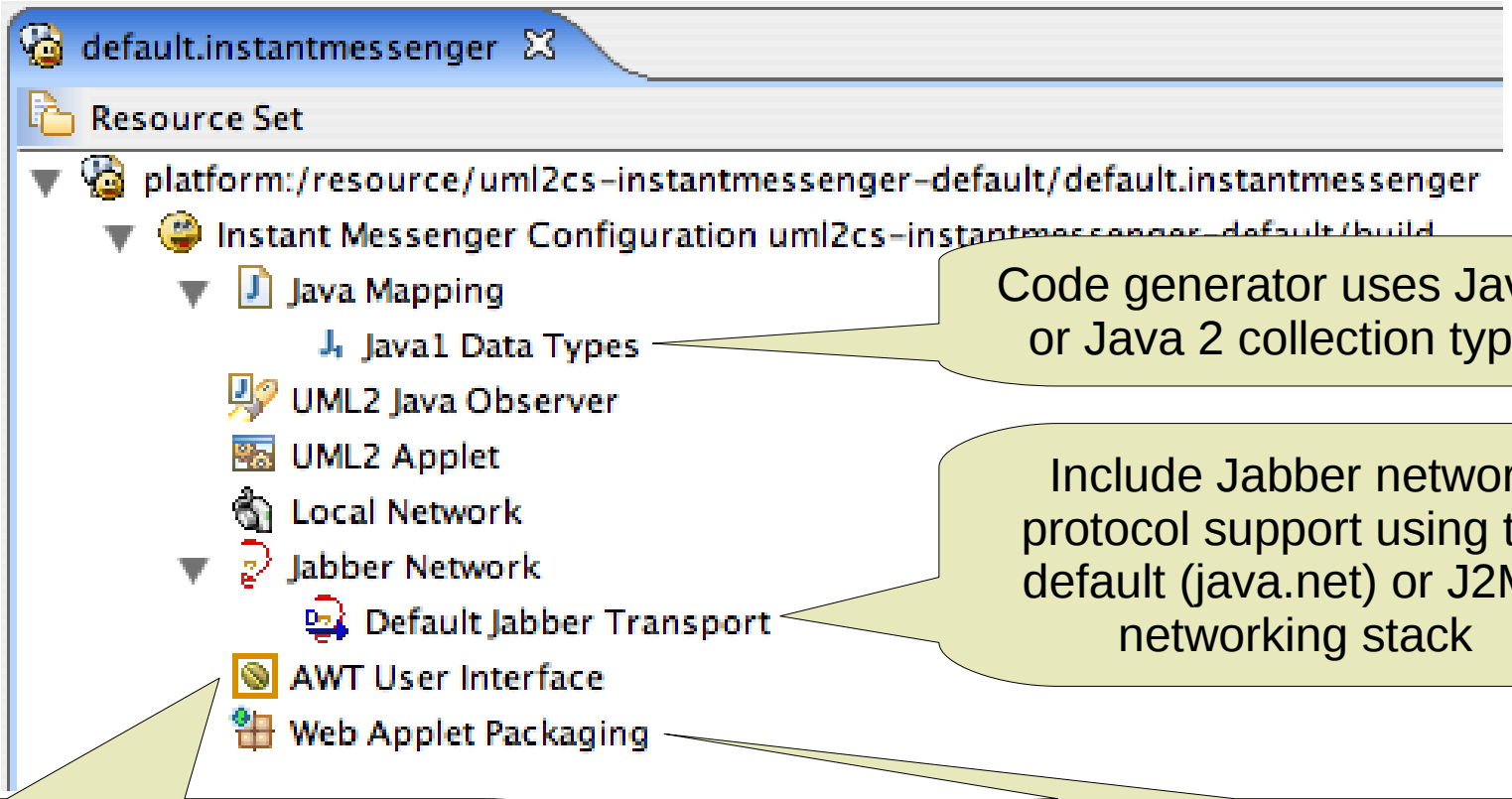
Contents

- Configuration languages
  - Example
- Interaction constraints vs. contextual constraints
  - Contextual constraint fragility
- Explicit context models
  - Using OWL DL ontologies
- Integrating contextual constraints in a configuration language
- Conclusions

Configuration languages

- Means to manage the variability of a software system, for example:
  - Product models for software product lines
  - Configuration files for software frameworks
- Can be defined in a variety of ways:
  - Grammar
  - XML schema
  - Meta-model

Configuration language example



Interaction constraints vs. contextual constraints (1)

- Generally, configuration constraints are interaction constraints, for example:
  - “When including Jabber network support, one must choose exactly one networking stack”
  - “One must choose at least one user interface”
- Some constraints have their cause in the context, however:
  - “When choosing to package as a MIDlet, one must choose the LCDUI user interface”
    - This constraint is caused by the fact that there is no operating context that supports AWT/Swing and MIDlets

Interaction constraints vs. contextual constraints (2)

- Interaction constraints are well-supported, and can be defined:
  - As part of the language's syntax, or
  - Separately, using a constraint language
- Expressing contextual constraints using standard methods gives rise to problems:
  - Constraints are expressed in terms provided by the configuration language
    - The context is not part of the language vocabulary
  - As a result, the context remains implicit

Contextual constraint fragility (1)

- Constraints that leave their context implicit are **fragile**: context evolves => constraint invalid?
  - “When choosing to package as a MIDlet, one must choose the LCDUI user interface”
  - Context evolves: new Java runtime comes along that supports MIDlets and AWT
  - The initial constraint no longer makes sense

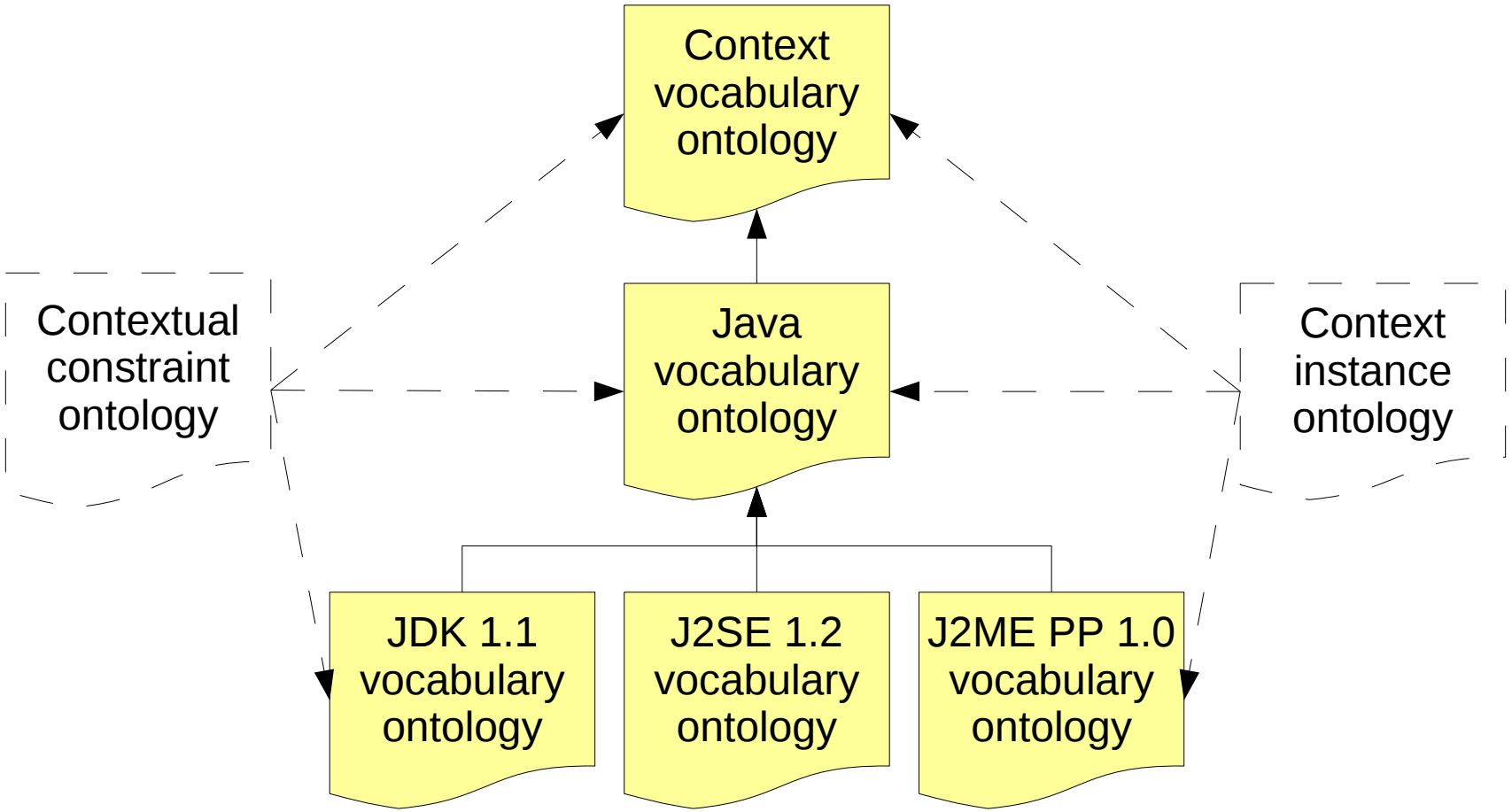
Contextual constraint fragility (2)

- Context must be **explicit** in contextual constraints!
  - “MIDlet packaging requires an MIDP Java runtime”
  - “The LCDUI user interface requires the Java runtime to provide the `javax.microedition.lcdui` API”
  - “The AWT user interface requires the Java runtime to provide the `java.awt` API”

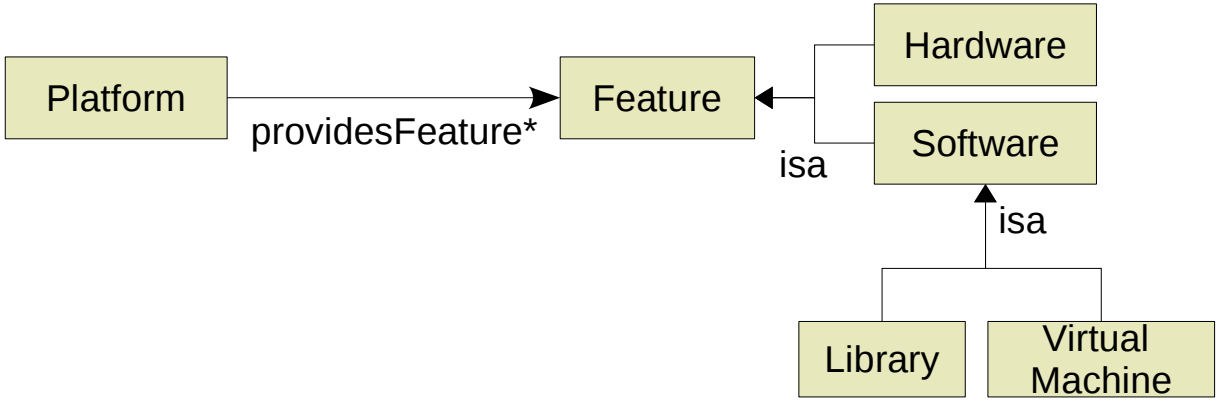
Explicit context models

- Provides vocabulary for expressing:
  - Contextual constraints in a configuration language
  - Context instances in which our system must operate
- Expressed as an OWL DL ontology
  - Ontologies have proven to be a suitable format for describing the concepts that can occur in the context

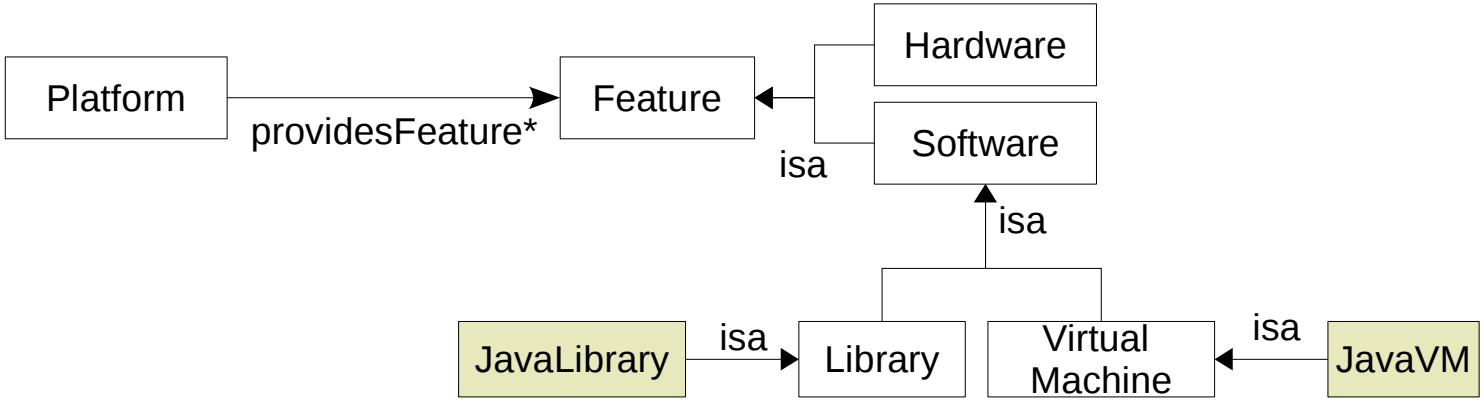
Context ontology



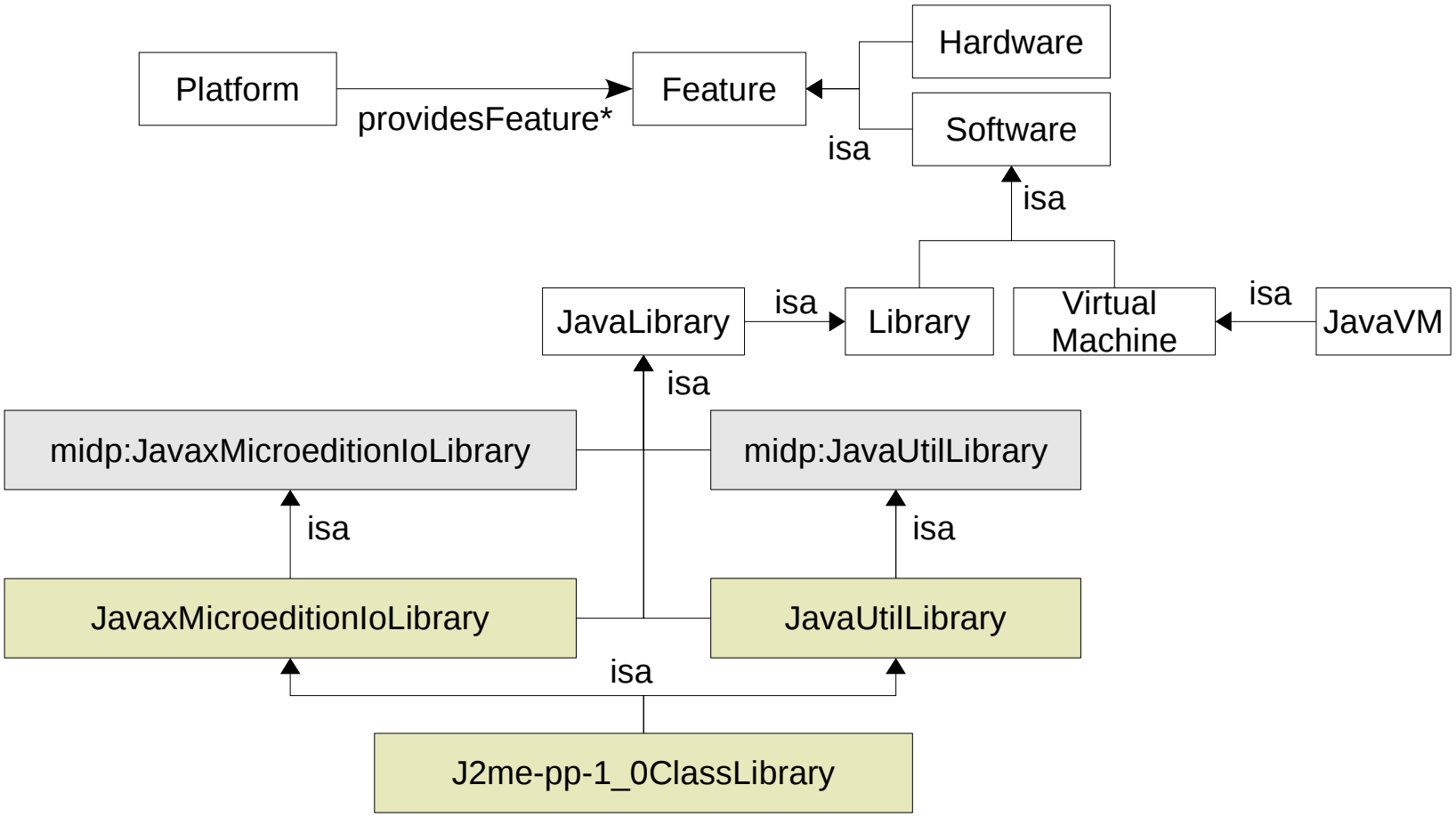
Context vocabulary ontology



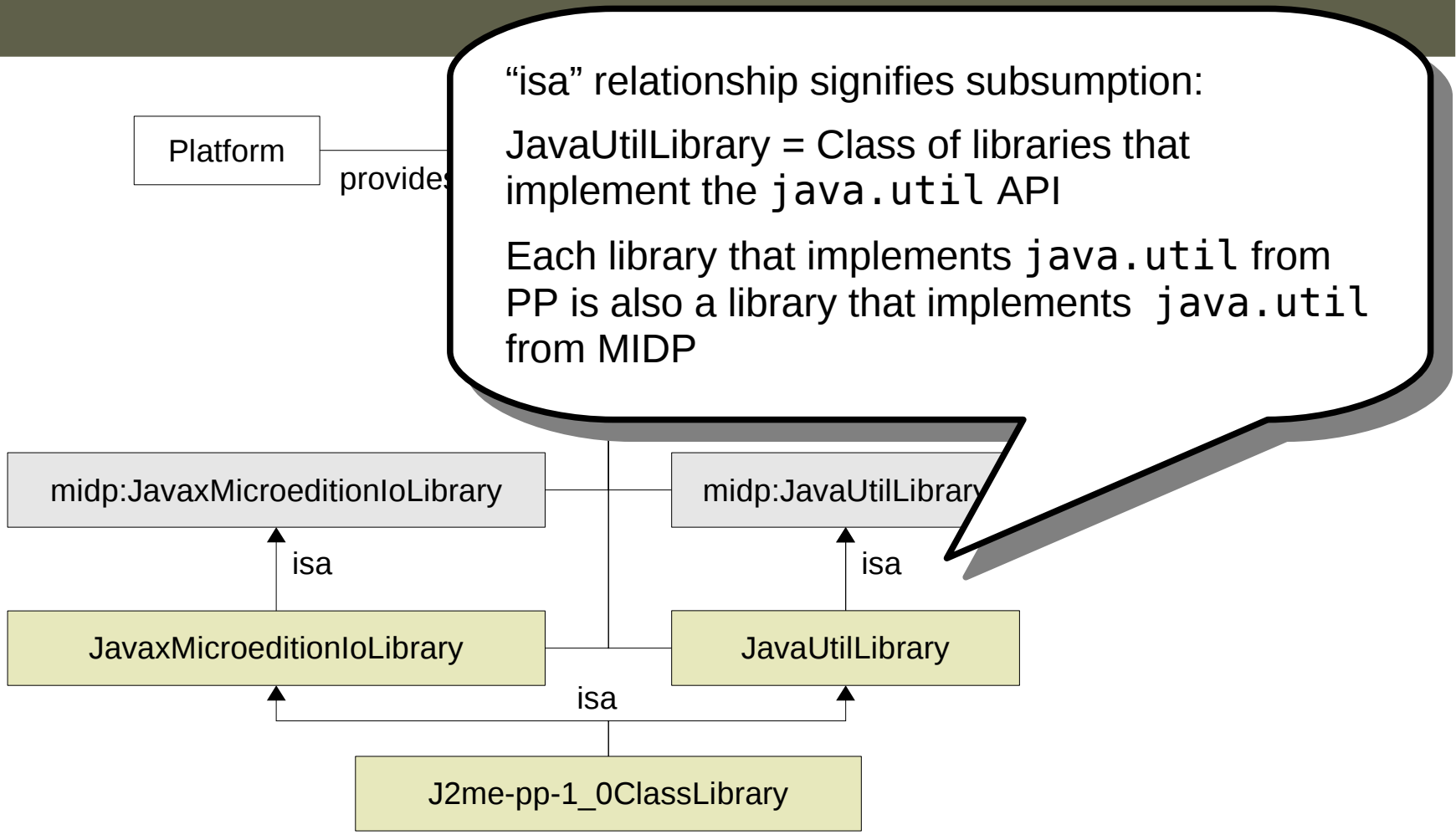
Java vocabulary ontology



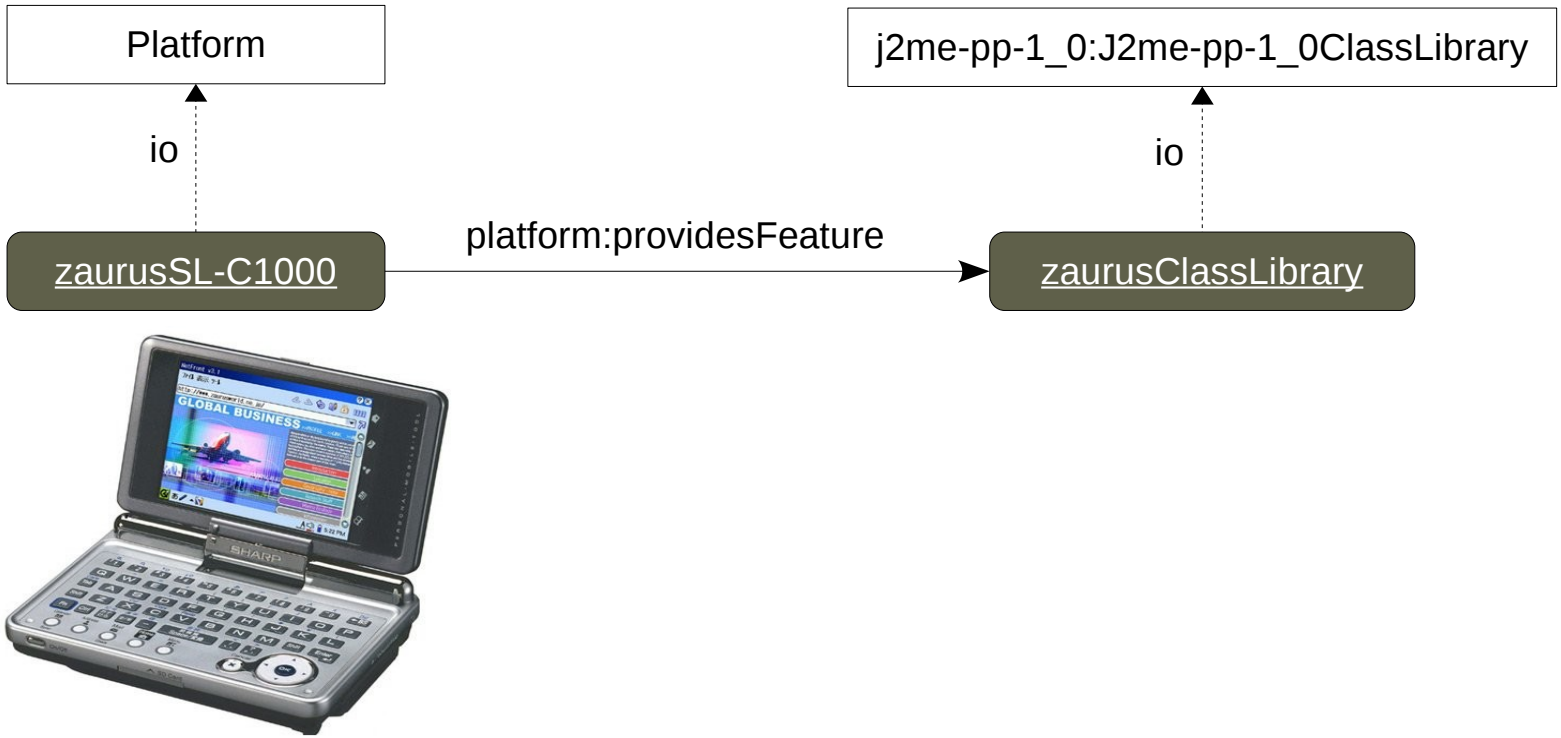
J2ME PP vocabulary ontology



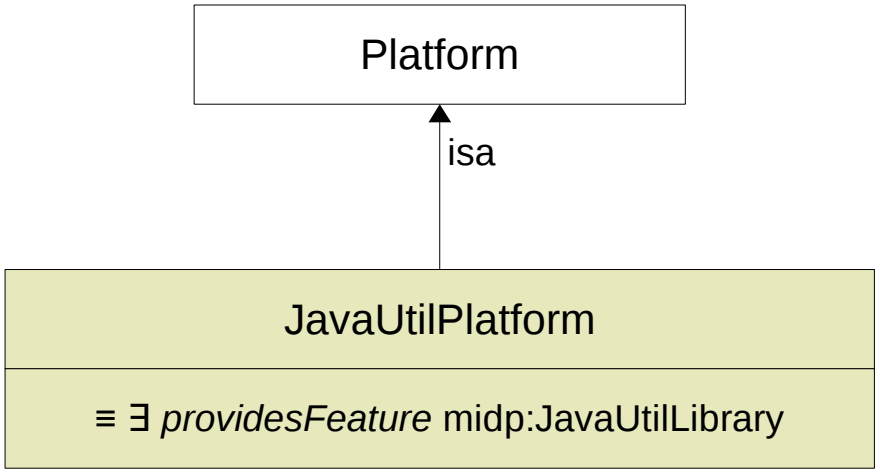
J2ME PP vocabulary ontology



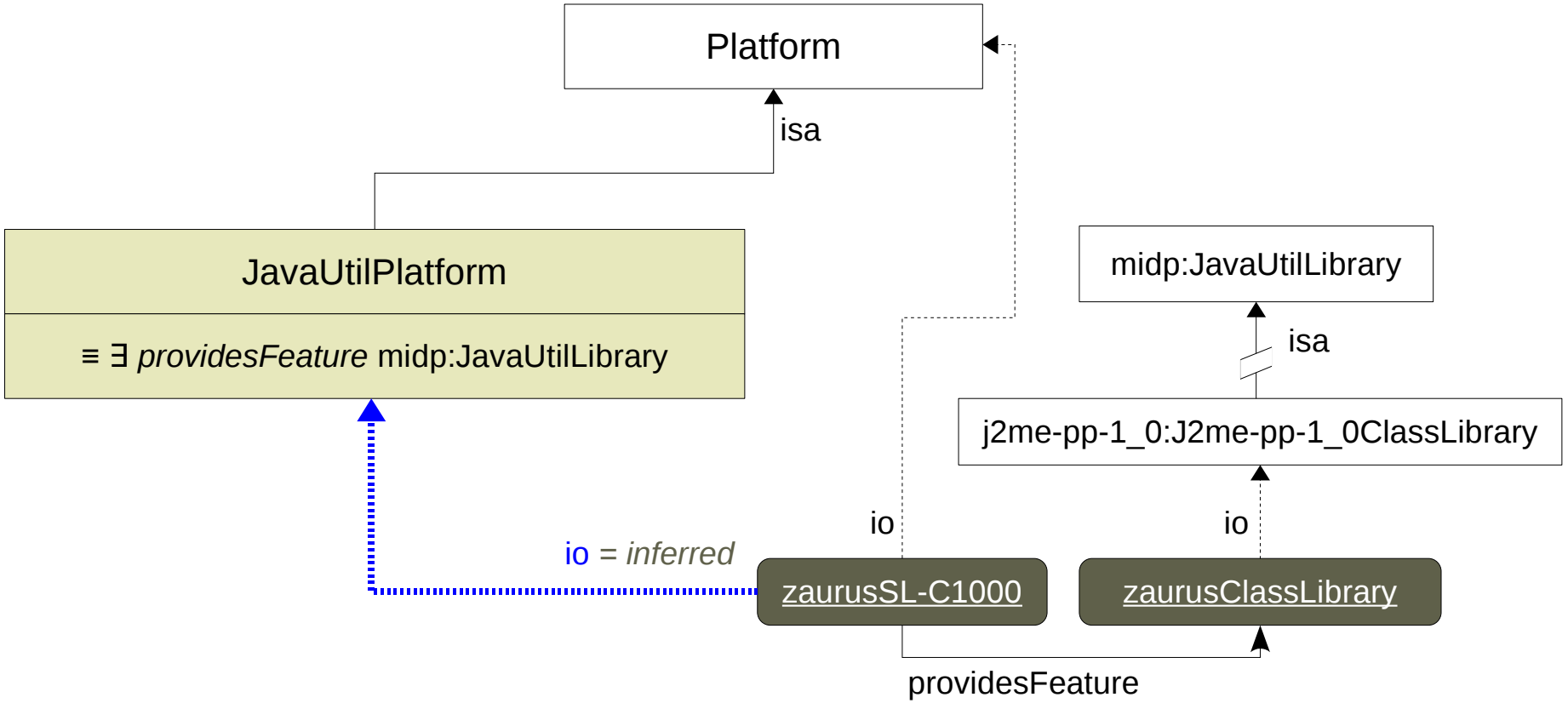
Context instances



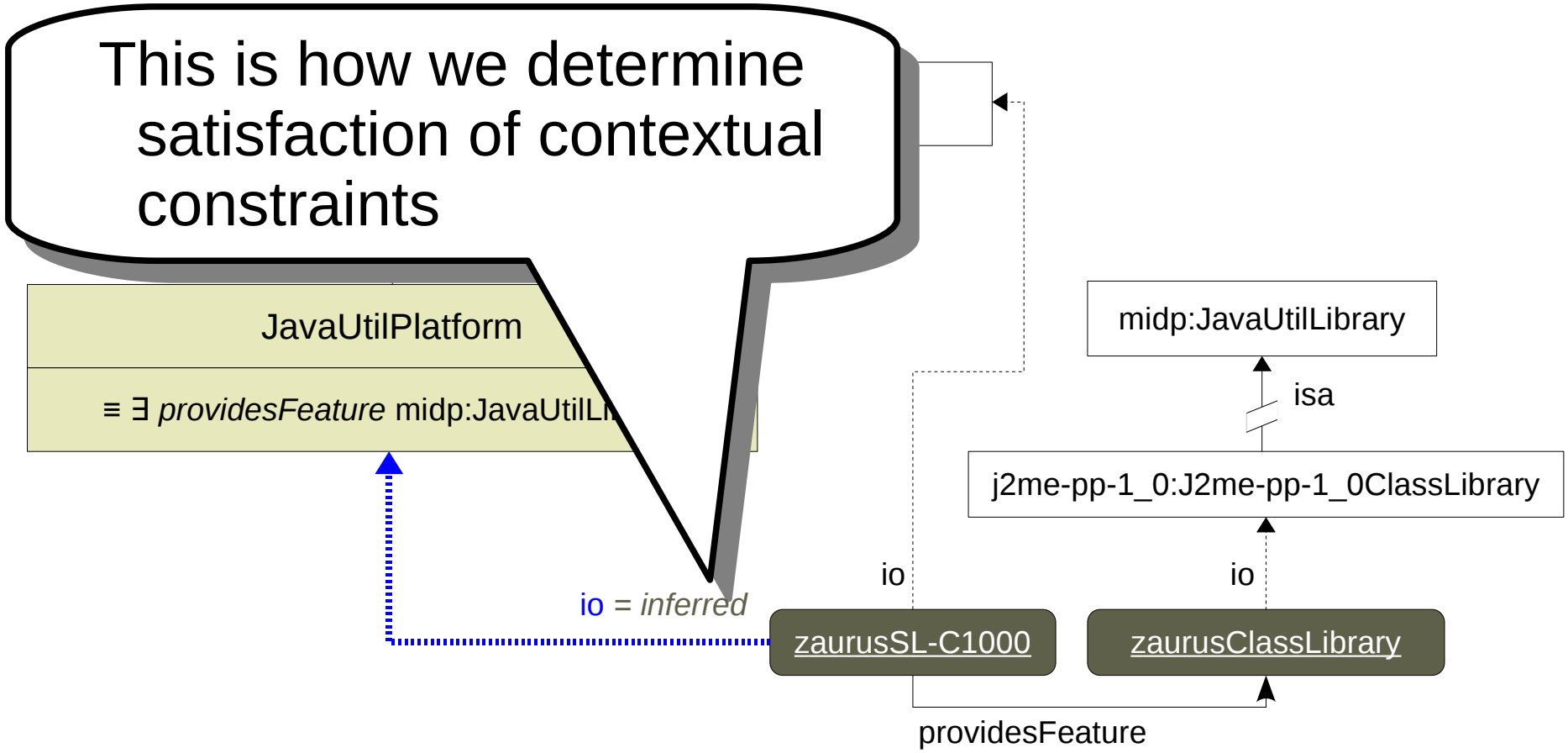
Contextual constraints



Contextual constraints



Contextual constraints



Contextual constraints

midp:JavaUtilLibrary

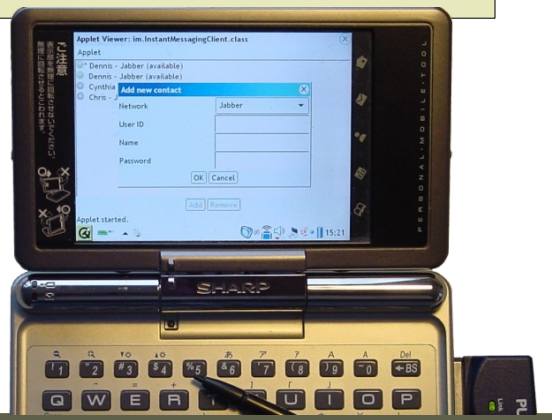
isa

pp:JavaUtilLibrary

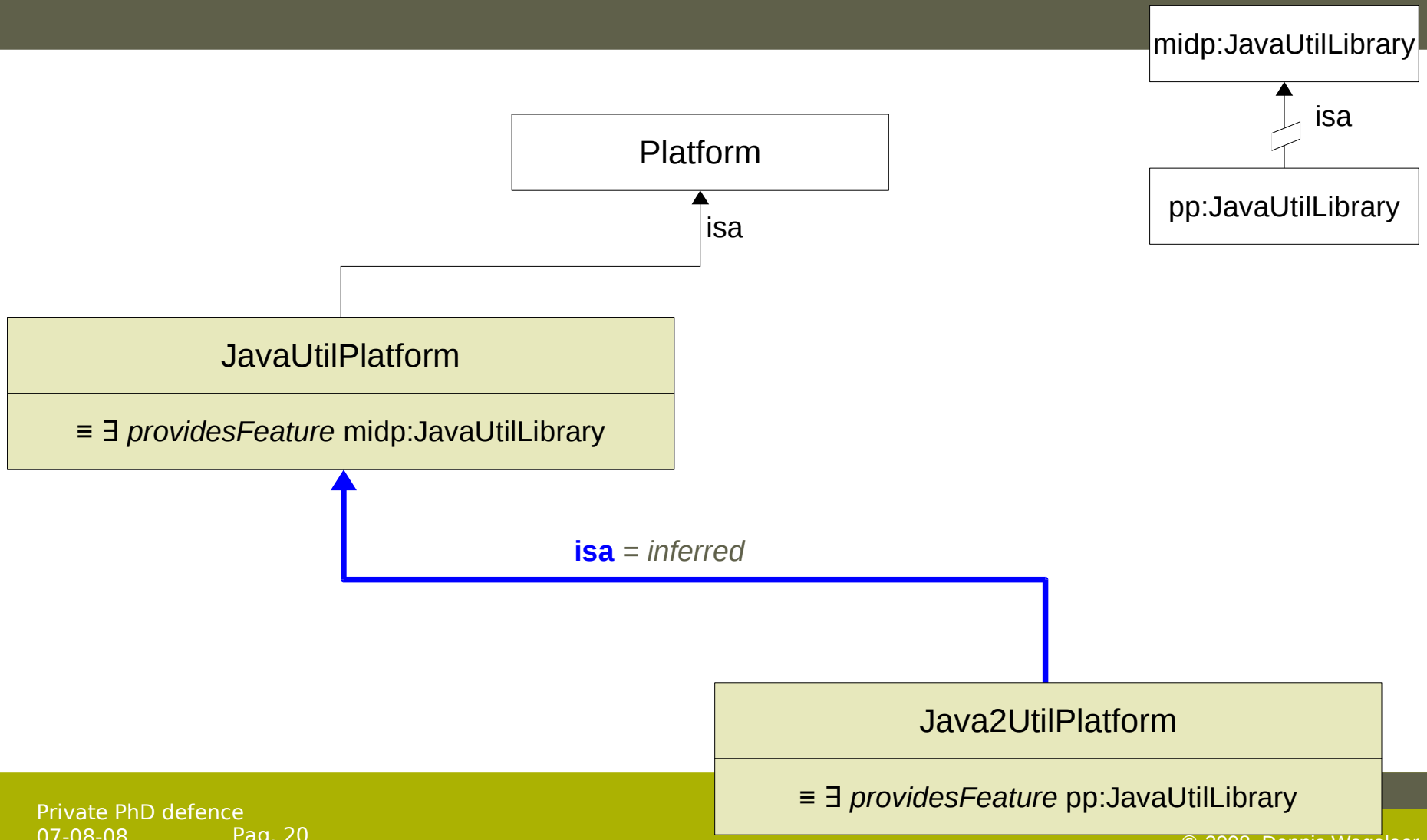
Platform

isa

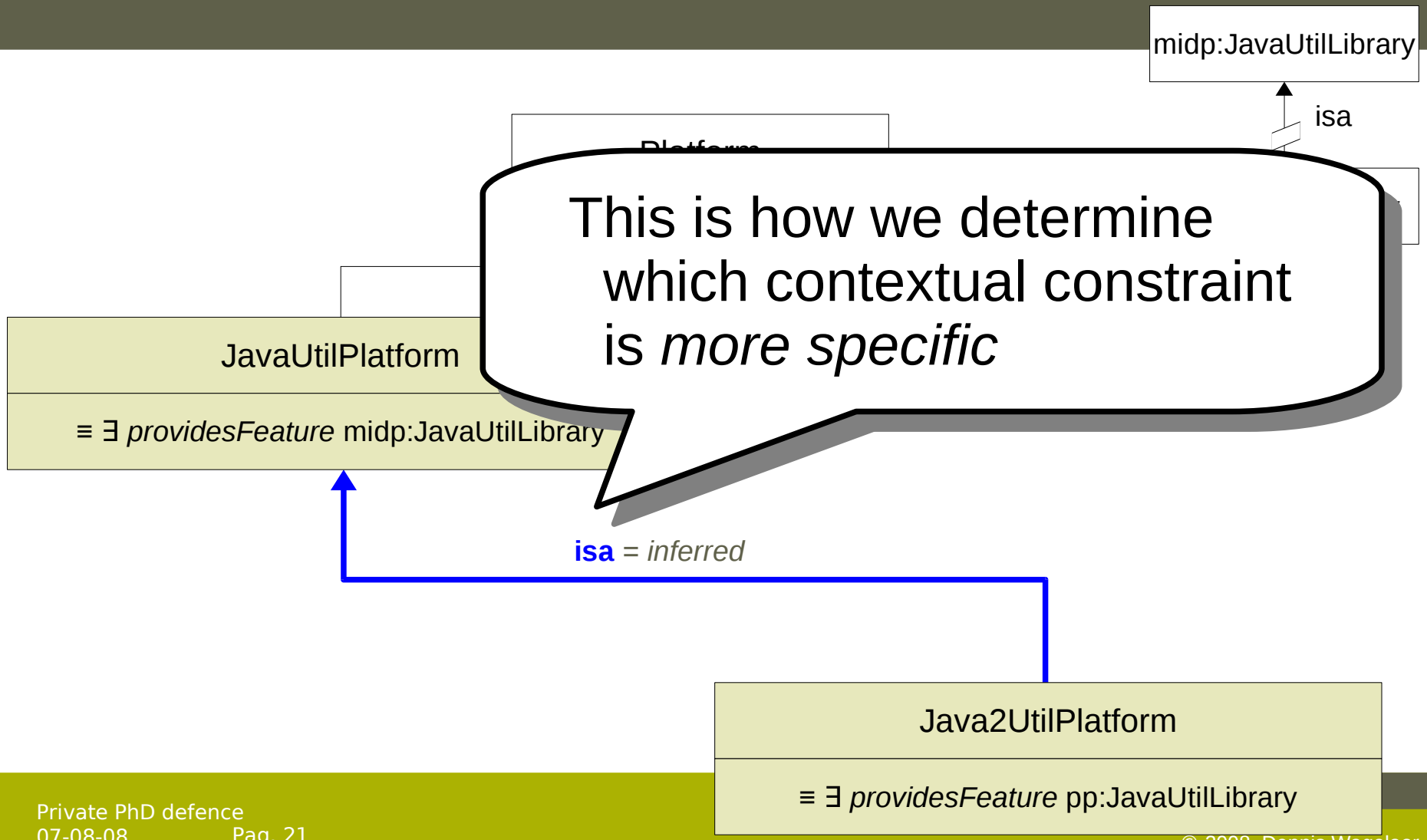
JavaUtilPlatform	Java2UtilPlatform
≡ ∃ *providesFeature* midp:JavaUtilLibrary	≡ ∃ *providesFeature* pp:JavaUtilLibrary



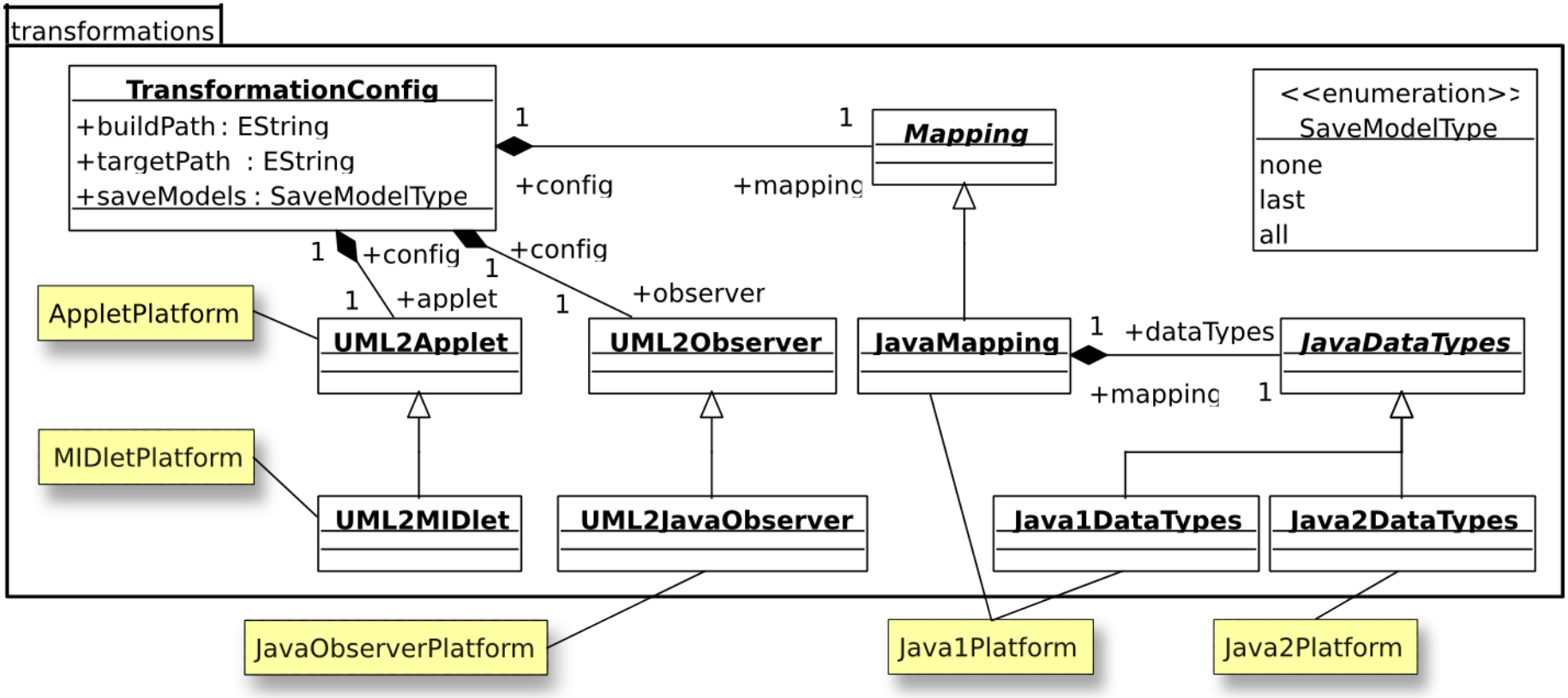
Contextual constraints



Contextual constraints



Integrating contextual constraints in a configuration language



Conclusions

- Contextual constraints are fragile when expressed as an interaction constraint
- Explicit context ontology reduces fragility of context constraints
- Context constraints can be used in the configuration process:
  - Eliminate invalid options for a given context
  - Optimisation for *most-specific* configuration option
- OWL DL reasoning performance is sufficient
  - Even for certain run-time configuration scenarios