Framework-Specific Modeling Languages with Round-Trip Engineering

Michal Antkiewicz & Krzysztof Czarnecki

University of Waterloo

Presented By:
Akshay Singh

Framework-Oriented Development

- Framework provides:
  - Concepts
  - Means of instantiating these concepts
- Framework Completion

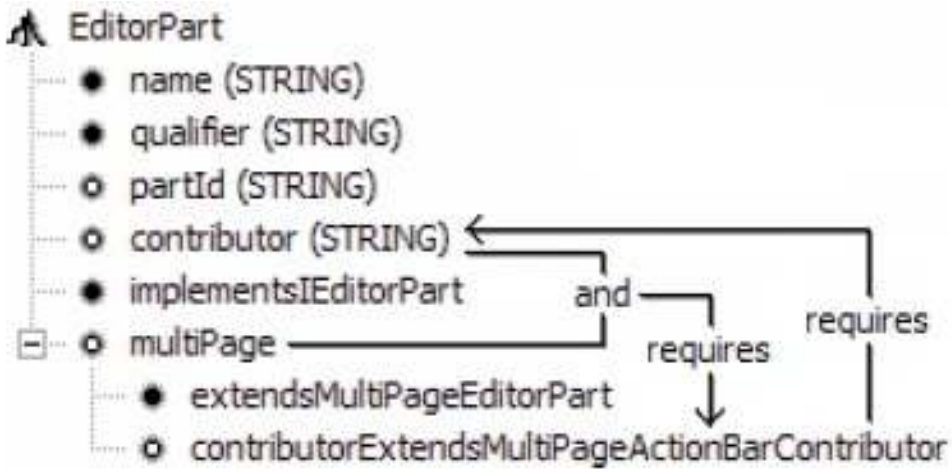
FSML

- Framework-Specific Modeling Language:
  - a kind of Domain-Specific Modeling Language
  - used for modeling framework-based software
  - Enables RTE over non-trivial model-code mapping

- express models showing how framework-provided concepts and their features are used in framework-based applications

Characterizing Framework Completion

- Concept configuration



- Open-Ended Programming with restriction

Challenges of Framework Completion

- Knowing how to complete a framework
- Obtaining application overview
- Following ‘rules of engagement’
- Repetitive code in concept instantiation
- Migration to evolved API

Addressing the Challenges

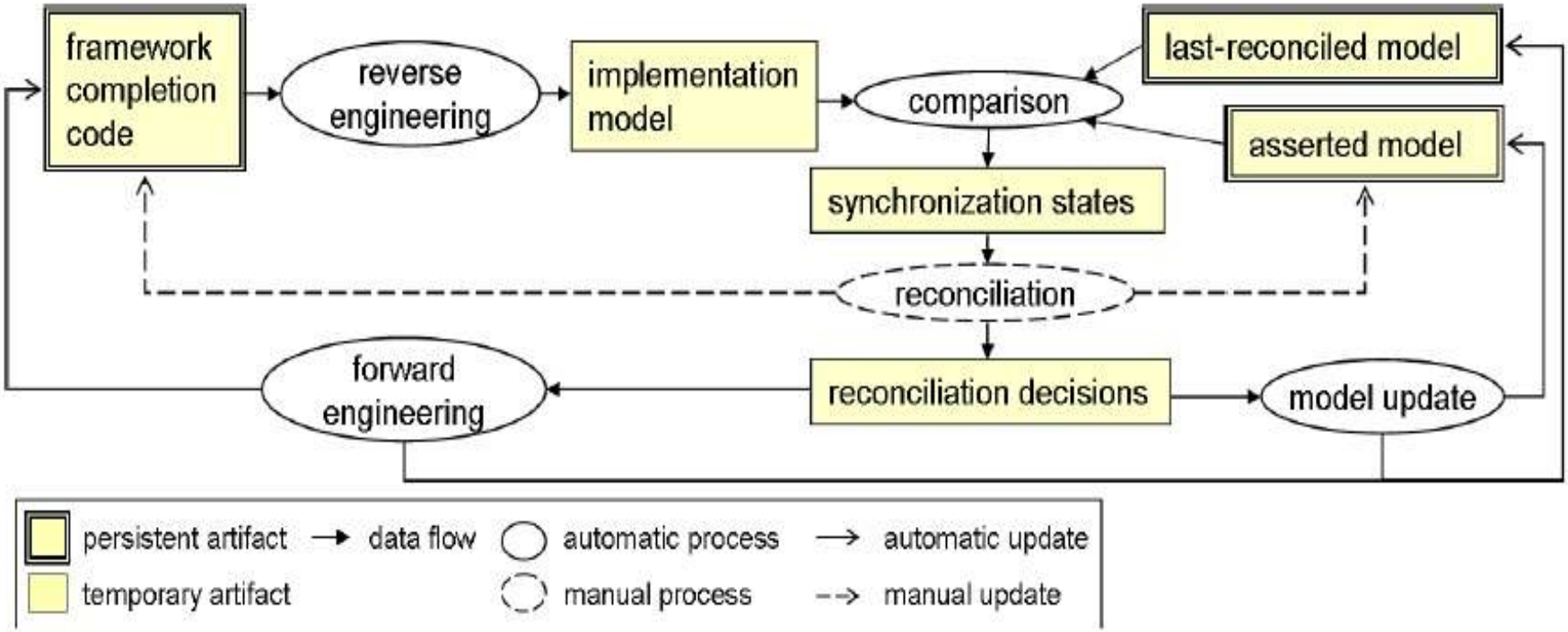
- FSML consists of:
  - Abstract syntax
  - Mapping of this syntax to framework API
  - Concrete syntax (optional)
- Round-Trip Engineering
  - Multi-directional synchronization

Agile RTE (1)

- FSMLs fill the abstraction gap between framework-provided concepts and framework-completion code

- Synchronization procedure involves:
  - Reverse Engineering
  - Comparison
  - Reconciliation
  - Forward Engineering

Agile RTE (2)


framework completion code
reverse engineering
implementation model
comparison
last-reconciled model
asserted model
synchronization states
reconciliation
forward engineering
reconciliation decisions
model update
persistent artifact
temporary artifact
data flow
automatic process
manual process
automatic update
manual update

Eclipse Workbench Part Interaction (WPI) – FSML (1)

Abstract Syntax

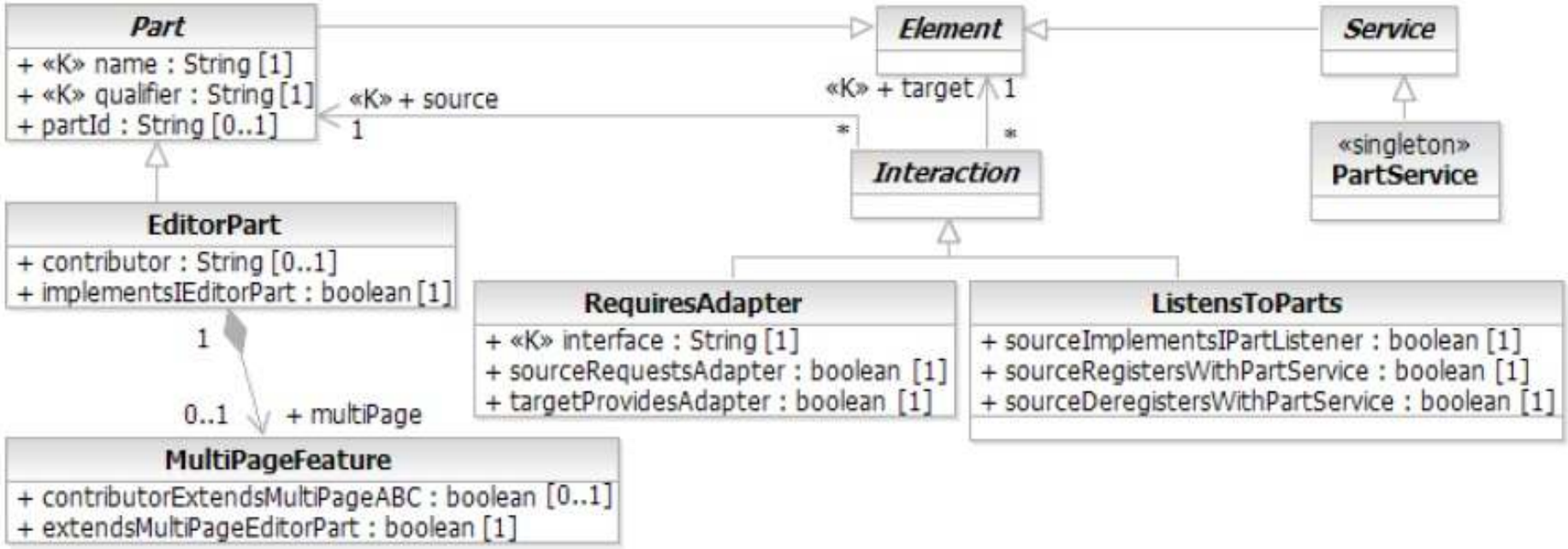


Fig. 3. Fragment of the metamodel of the WPI FSML expressed in MOF

WPI – FSML (2)

Mapping abstract syntax to the framework API

mapping EditorPart(EditorPart ep <-> Class editor);

- key name

  ←ep.name = editor.name;

  → RENAME(editor, ep.name);

- optional partId

  ←ep.partId = EDITORID(editor);

  → EDITORID(ep.qualifier + "." + ep.name, ep.partId);

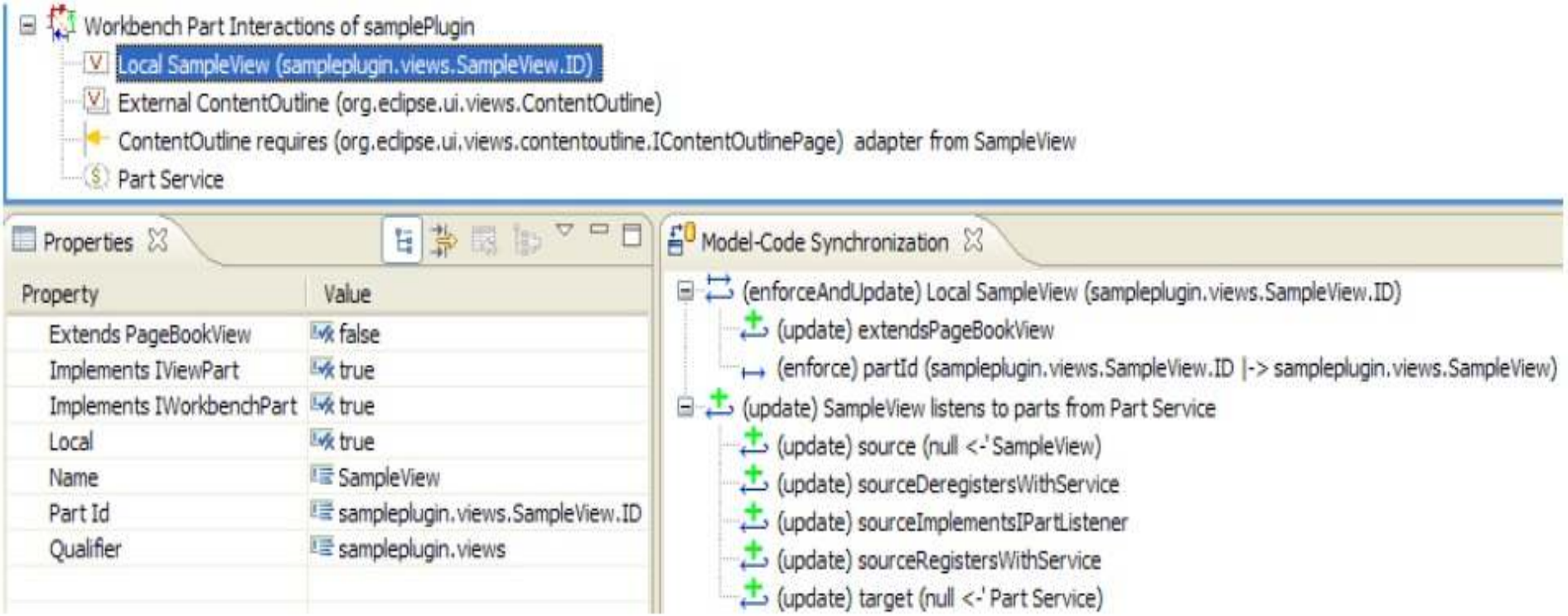
WPI - Prototype

WPI – FSML captures an aspect of Eclipse plug-in development.

Steps involved:

- create a sample Eclipse plugin using wizard
- reverse-engineer the plug-in code to create its WPI model
- modify and synchronize

WPI - Model


Workbench Part Interactions of samplePlugin
Local SampleView (sampleplugin.views.SampleView.ID)
External ContentOutline (org.eclipse.ui.views.ContentOutline)
ContentOutline requires (org.eclipse.ui.views.contentoutline.IContentOutlinePage) adapter from SampleView
Part Service
Properties
Property
Value
Extends PageBookView
false
Implements IViewPart
true
Implements IWorkbenchPart
true
Local
true
Name
SampleView
Part Id
sampleplugin.views.SampleView.ID
Qualifier
sampleplugin.views
Model-Code Synchronization
(enforceAndUpdate) Local SampleView (sampleplugin.views.SampleView.ID)
(update) extendsPageBookView
(enforce) partId (sampleplugin.views.SampleView.ID |-> sampleplugin.views.SampleView)
(update) SampleView listens to parts from Part Service
(update) source (null <-' SampleView)
(update) sourceDeregistersWithService
(update) sourceImplementsIPartListener
(update) sourceRegistersWithService
(update) target (null <-' Part Service)

Conclusion

- Paper proposes the concept of FSML with full round-trip engineering (fine-grained mapping)
- Addresses many challenges of framework completion
- Enable developers to modify application code
- Presented implementation prototype giving a proof of concept, though quite elementary.

References

- Antkiewicz, M., and K. Czarnecki, "*Framework-Specific Modeling Languages with Round-Trip Engineering*", ACM/IEEE 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS), Genova, Italy, Springer-Verlag, pp. 692-706, 2006.

- Antkiewicz, M., and K. Czarnecki, "*Round-Trip Engineering of Eclipse Plug-Ins Using Eclipse Workbench Part Interaction FSML",* OOPSLA'06 October 22–26, 2006, Portland, Oregon, USA.

Discussion

- Forward mappings may need some orchestration mechanism, in case of sophisticated dependencies among artifacts.
- Requires explicit annotations for reverse engineering, so manual changes to the code would not be natural.
- No direct support for call graphs and include dependencies.
- A single FSML would cover a small functionalities of Framework!

Thanks!