FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Diplomarbeitspräsentation

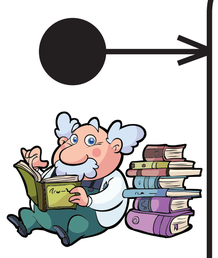
Integration of External Libraries into the Foundational Subset of UML

Masterstudium:
Wirtschaftsinformatik

Patrick Neubauer

Technische Universität Wien
Institut für Softwaretechnik und Interaktive Systeme
Arbeitsbereich: Business Informatics Group
BetreuerIn: O.Univ.Prof. Dipl.-Ing. Mag. Dr.techn. Gerti Kappel

Context



Motivation

- Merging the benefits of Model Driven Engineering (MDE) and software reusability.
  - MDE: Increased productivity, portability, interoperability, maintainability, and organizational benefits [1].
  - Software reusability: Rapid creation of applications performing advanced and complex tasks building on well-tested existing software libraries that improve the overall quality and productivity [2].

Problem Statement

- fUML virtual machine does **not** allow using external libraries.
- fUML standard foresees extendibility through the Foundational Model Library [3].
  - Drawback: Requires a huge amount of dedicated joint effort to equipt this library with the same power as today's Java libraries.
  - Hence, models that perform tasks of today's applications cannot be quickly built.

► **Is it possible to overcome the unability of the fUML virtual machine to access functionality of existing libraries?**

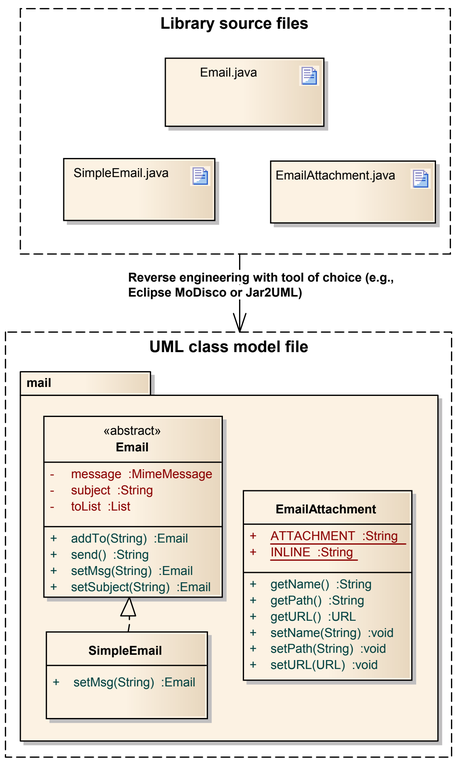
Problem

Our Approach

Problem

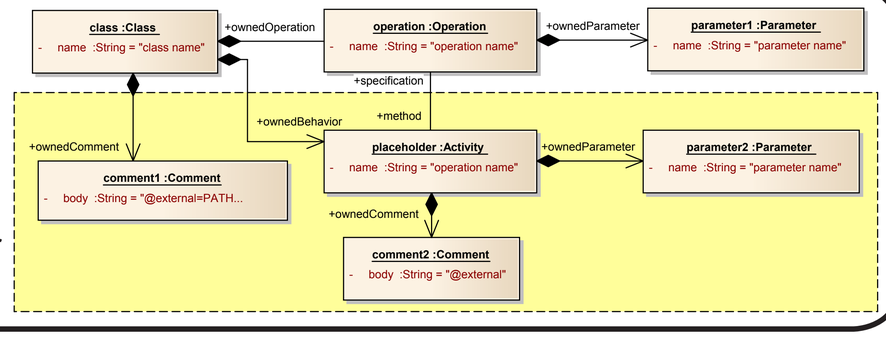
Step 1: Reverse Engineering

- Today's reverse engineering tools can only reverse engineer structure.
- Therefore, Java source code is reverse engineered into a UML class model representing the external library's **structure** (i.e., **what** the library contains).



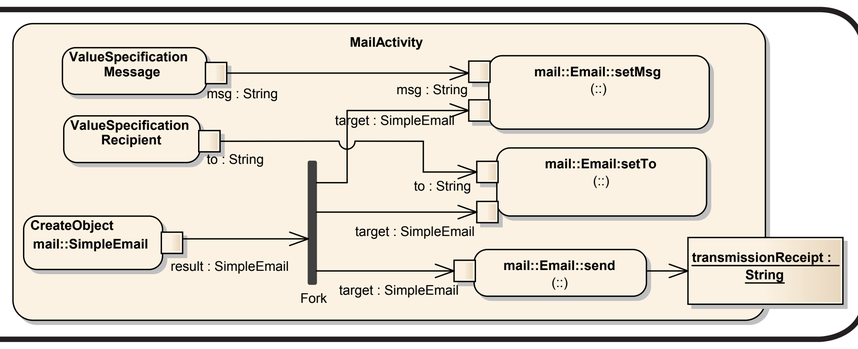
Step 2: Preparing

- The library's **behavior** (i.e., **how** an operation performs) is contained in the operation's body.
- By adding additional information to the UML class model from step 1, behavior can be located during execution time.



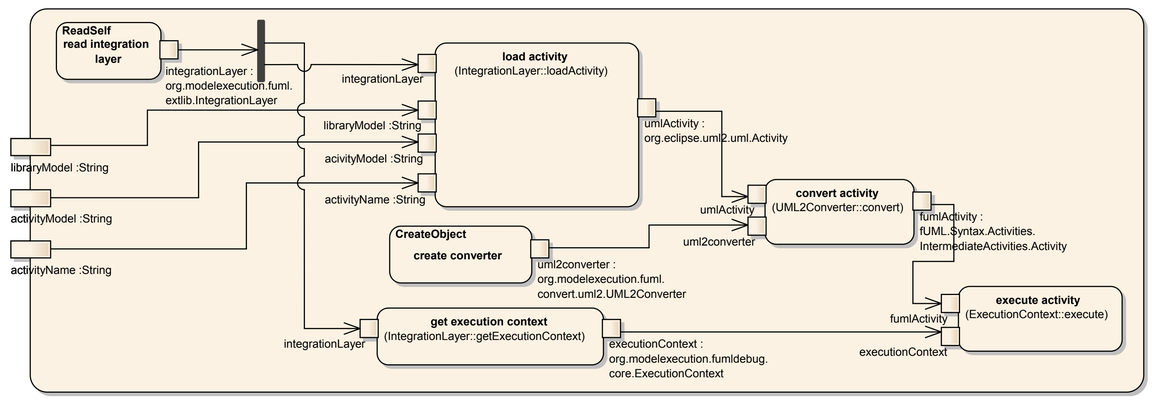
Step 3: Modeling

- The UML activity references the prepared UML class model from step 2.
- In the example, the `classifier` attribute of the `CreateObjectAction` references the ApacheMail library's `SimpleEmail` class.



Step 4: Executing

- Library behavior is accessed by using Java Reflection on the library's JAR file.
- The UML activity from step 3 is first converted into an fUML activity and then used for execution.
- The Integration Layer listens on any occurring event and acts in case external library access is required.
- Parameters are translated bidirectionally between fUML and Java.



Our Solution

Evaluation and Results

Our Solution

Evaluation

- Concerning usability, correctness, and performance.
- Evaluated within three case studies.
- How does the prototype fit into the overall modeling process?
- Comparing expected and produced outcome.
- Performance comparison by running both the activity model and a plain Java application behaving the same way.

Results

- **Usability:** Prototype does not affect overall modeling process. Unsupported scenarios have been identified.
- **Correctness:** Case studies produced the expected result.
- **Performance:** No significant execution time difference between activity model and plain Java application execution.

► **It is possible to overcome the limitation or unability to access functionality of existing libraries by integrating them with the fUML virtual machine using a dedicated Integration Layer.**

References:
[1] John Hutchinson, Jon Whittle, Mark Rouncefield, and Steinar Kristoffersen. Empirical Assessment of MDE in Industry. In Proceedings of the 33rd International Conference on Software Engineering, ICSE '11, pages 471–480, New York, NY, USA, 2011. ACM.
[2] Suryani Ismail, Wan M.N. Wan-Kadir, Yazid M. Saman, and Siti Z. Mohd-Hashim. A review on the component evaluation approaches to support software reuse. In International Symposium on Information Technology, volume 4, pages 1–6, 2008.
[3] Object Management Group. Semantics of a Foundational Subset for Executable UML Models (fUML), Version 1.1, 2012.

Kontakt: pneubauer@acm.org