Model-based Analysis of Source Code Repositories

Markus Scheidgen   Martin Schmidt   Joachim Fischer

{scheidge,schmidma,fischer@informatik.hu-berlin.de}
Humboldt Universität zu Berlin
Agenda

- *Software Evolution, Reverse Engineering, and Mining Software Repositories*
- *Model-based Mining of Software Repositories*
- *srcrepo* a framework for model-bases analysis of software repositories
- Experiments with *Eclipse*’s software repositories
- Conclusions
Software Evolution

software engineering

requirements

user

software maintenance

problem

user

software modernization

transformation

OMG’s ADM
(Architecture-Driven Modernization)
- AST Meta-Model (ASTM)
- Knowledge Discovery Meta-Model (KDM)
- Software Metrics Meta-Model (SMM)

mining software repositories

quality assessment
implicit dependencies

Mining Software Repositories – In General

▶ The term mining software repositories (MSR) has been coined to describe a broad class of investigations into the examination of software repositories.

▶ The premise of MSR is that empirical and systematic investigations of repositories will shed new light on the process of software evolution. [1]

▶ Different scopes, e.g. single software projects vs. many software projects

▶ Different goals, e.g. quality assessments and implicit dependencies vs. generalizations about software evolution

Model-based Mining of Software Repositories

Model-based Mining of Software Repositories

Model-based Mining of Software Repositories

Model-based Mining of Software Repositories

Model-based Mining of Software Repositories

reverse engineering

Model-based Mining of Software Repositories

Scope

- depends on concreter MSR-application and its goals
- number of software projects: single repositories, large repositories, ultra-large repositories
- Sources as text and text based metrics, e.g. LOC
- Declarations only: packages, classes, methods, but no statements, expressions, etc.
- Full AST with or without cross-references
Model-based Mining of Software Repositories

Scope

- depends on concreter MSR-application and its goals
- number of software projects: single repositories, large repositories, ultra-large repositories
- Sources as text and text based metrics, e.g. LOC
- Declarations only: packages, classes, methods, but no statements, expressions, etc.
- Full AST with or without cross-references
Model-based Mining of Software Repositories

- MSR tools are already “model-based”, but in a proprietary manner

- *Idea*: existing reverse engineering framework and corresponding standard meta-models and modeling frameworks instead of proprietary solutions

- Goals
  - deal with heterogeneity (different version control systems, different languages)
  - reuse of existing meta-models, transformations, and languages
  - interoperability with existing analysis tools
  - retaining meaningful scalability
Model-based MSR Strategies
Model-based MSR Strategies

CHECKOUT($r$)
Model-based MSR Strategies

\textbf{CHECKOUT}(r)

`version control system`

snapshot
\[ A_2 \quad B_1 \]

snapshot
\[ A_1 \quad B_1 \]

`View of a version control system`
Model-based MSR Strategies

CHECKOUT(r)

version control system

A_1, A_2

A_1

B_1

B_1, B_3
Model-based MSR Strategies

\[ \sum_{d \in \text{CUs}(r)} \text{PARSE}(d) \]

**Checkout** \((r)\)

version control system

A\(_1\), A\(_2\), A\(_1\) - A\(_2\), B\(_1\), B\(_1\) - B\(_3\)

snapshot

A\(_1\), B\(_1\)

A\(_2\), B\(_1\)

A\(_2\), B\(_3\)

snapshot

X\(_d\), CUs\((r)\)

Parse \((d)\)

snapshot
Model-based MSR Strategies

\[ \sum_{d \in \text{CUs}(r)} \text{PARSE}(d) \]

Checkout \((r)\)
Model-based MSR Strategies

\[ \sum_{d \in \text{CUS}(r)} \text{PARSE}(d) \]

\text{CHECKOUT}(r)
Model-based MSR Strategies

\[ \sum_{d \in \text{CUs}(r)} \text{PARSE}(d) \rightarrow \text{ANALYSIS}(r) \rightarrow M_1 \]

**CHECKOUT**(r)

version control system

\[ A_1 - A_2, B_1 - B_3 \]
Model-based MSR Strategies

\[ \sum_{d \in \text{CUs}(r)} \text{PARSE}(d) \rightarrow \text{CHECKOUT}(r) \rightarrow \text{ANALYSIS}(r) \]

version control system

A_1-A_2

A_1

B_1

B_1-B_3

A_2

B_3

M_1

M_2
Model-based MSR Strategies

\[
\sum_{d \in \text{CU}s(r)} \text{PARSE}(d)
\]

CHECKOUT\((r)\)

version control system

\[
\begin{align*}
\text{A}_1 - \text{A}_2 & \quad & \text{B}_1 - \text{B}_3 \\
\text{A}_1 & \quad & \text{B}_1
\end{align*}
\]
Model-based MSR Strategies

**CHECKOUT**

- **A_1**
- **B_1**

**version control system**

\[
\sum_{d \in \text{CUs}(r)} \text{PARSE}(d)
\]

\[
\sum \left( \text{CHECKOUT} + \sum_{\text{CUs}} \text{PARSE} + \text{ANALYSIS} \right)
\]
Model-based MSR Strategies

\[ \sum_{d \in CUs(r)} \text{PARSE}(d) \]

\( \text{CHECKOUT}(r) \)

\( \text{VERSION CONTROL SYSTEM} \)

\( A_1, A_2, B_1, B_3 \)

\( M_1, M_2, M_3 \)

\( \text{ANALYSIS}(r) \)
Model-based MSR Strategies

CHECKOUT\((r)\)

\[ \sum_{d \in CUs(r)} \text{PARSE}(d) \]

\[ \sum_{d \in \Delta CUs(r)} \text{PARSE}(d) \]

version control system

\[ A_1, A_2, B_1, B_3, B_1, B_3 \]

\[ A_2, B_1 \]

\[ A_1, B_1 \]

snapshot

\[ A_2, B_3 \]

\[ A_2, B_1 \]

\[ A_1, B_1 \]

\[ M_1, M_2, M_3 \]

\[ M_3 \]

\[ M_2 \]

\[ M_1 \]

\[ f \]

\[ B.f \]

\[ B.f \]

\[ f \]
Model-based MSR Strategies

CHECKOUT($r$)

$$\sum_{d \in \text{CUS}(r)} \text{PARSE}(d)$$

$$\sum_{d \in \Delta\text{CUS}(r)} \text{PARSE}(d)$$

ANALYSIS($r$)

MERGE($r$)
Model-based MSR Strategies

\[
\sum_{d \in \text{CUs}(r)} \text{Parse}(d) \\
\sum_{d \in \Delta \text{CUs}(r)} \text{Parse}(d)
\]

\[
\text{CHECKOUT}(r) \rightarrow \text{PARSE}(d) \rightarrow \text{ANALYSIS}(r) \rightarrow \text{MERGE}(r)
\]

\[
\sum_R \left( \text{CHECKOUT} + \sum_{\text{CUs}} \text{PARSE} + \text{ANALYSIS} \right)
\]

\[
\gg \sum_R \left( \text{CHECKOUT} + \sum_{\Delta \text{CUs}} (\text{PARSE} + \text{MERGE}) + \text{ANALYSIS} \right)
\]
Model-based MSR Strategies

Checkout \( (r) \)

\[ \sum_{d \in \text{CUs}(r)} \text{PARSE}(d) \]

\[ \sum_{d \in \Delta \text{CUs}(r)} \text{PARSE}(d) \]

Analysis \( (r) \)

Merge \( (r) \)
Model-based MSR Strategies

\[ \sum_{d \in \text{CUs}(r)} \text{PARSE}(d) \]

\[ \sum_{d \in \Delta \text{CUs}(r)} \text{PARSE}(d) \]

\[ \text{CHECKOUT}(r) \]

\[ \text{LOAD}(r) \]

\[ \text{SAVE}(r) \]

\[ \text{MERGE}(r) \]

\[ \text{ANALYSIS}(r) \]
\[
\sum_{R} \left( \text{CHECKOUT} + \sum_{CUs} \text{PARSE} + \text{ANALYSIS} \right)
\]
\[
\gg \sum_{R} \left( \text{CHECKOUT} + \sum_{\Delta CUs} (\text{PARSE} + \text{MERGE}) + \text{ANALYSIS} \right)
\]
\[
\gg \sum_{R} \left( \sum_{\Delta CUs} (\text{LOAD} + \text{MERGE}) + \text{ANALYSIS} \right)
\]
\[ \sum_{R} \left( \text{CHECKOUT} + \sum_{\text{CUs}} \text{PARSE} + \text{ANALYSIS} \right) \]

\[ \gg \sum_{R} \left( \text{CHECKOUT} + \sum_{\Delta\text{CUs}} (\text{PARSE} + \text{MERGE}) + \text{ANALYSIS} \right) \]

\[ \gg \sum_{R} \left( \sum_{\Delta\text{CUs}} (\text{LOAD} + \text{MERGE}) + \text{ANALYSIS} \right) \]
Model-based MSR Strategies

\[
\sum_R \left( \text{CHECKOUT} + \sum_{\text{CUs}} \text{PARSE} + \text{ANALYSIS} \right)
\]

\[
\gg \sum_R \left( \text{CHECKOUT} + \sum_{\Delta \text{CUs}} \left( \text{PARSE} + \text{MERGE} \right) + \text{ANALYSIS} \right)
\]

\[
\gg \sum_R \left( \sum_{\Delta \text{CUs}} \left( \text{LOAD} + \text{MERGE} \right) + \text{ANALYSIS} \right)
\]

\[
\gg \sum_R \sum_{\Delta \text{CUs}} \left( \text{LOAD} + \text{ANALYSIS}' \right)
\]
Model-based MSR Strategies

\[ \text{CHECKOUT}(r) \]

\[ \text{version control system} \]

\[ \text{snapshot} \]

\[ \begin{align*}
A_2 & \quad \quad B_3 \\
A_2 & \quad \quad B_1 \\
A_1 & \quad \quad B_1
\end{align*} \]

\[ \text{snapshot} \]

\[ \begin{align*}
\sum_{d \in \text{CUs}(r)} \text{Parse}(d) & \\
\sum_{d \in \Delta \text{CUs}(r)} \text{Merge}(d) & \\
\text{SIS}(r) & \\
\text{E}(r) & \\
\text{ANALYSIS}'(r)
\end{align*} \]
Research Questions

Assumptions

- Development of MSR-applications based on models, transformation languages and standardized meta-models is favorable.
- Some MSR-applications need to analyze source code on a deep (AST) level.
- MSR-analysis is performed iteratively.

Hypotheses

- Models of source code repositories can be created and persisted.
- Traversing existing persistent models of source code repositories is much faster than traversing transient models that are created from version control system on the fly.
srcrepo – A Framework for Model-based MSR

- Eclipse’s MoDisco as reverse engineering framework
  - reverse engineering for Java, based on EMF
  - Support for many JRE-based languages: Java, xText, JSP, XML
  - creates instances of a Java EMF meta-model that corresponds to the handwritten JDT AST-model
  - provides transformation to language independent artifacts, e.g. KDM

- EMF-Fragments\(^1\) to store very large-models
  - uses No-SQL databases and stores larger model fragments within database entries
  - in contrast to object-by-object stores such as ORM-based CDO or No-SQL-based Morsa or Neo4J

- Xtend programming with higher order functions to mimic OCL-style definition of software metrics\(^2\)

---

“OCL” to Calculate Metrics of AST-Models

// Weighted number of methods per class.
def wmc(AbstractTypeDeclaration type,(Block)⇒int weight) {
    type.bodyDeclarations.sum[if (body != null) weight.apply(it.body) else 1]
}
Experiments

- Eclipse Foundation sources, i.e. Eclipse platform and plug-ins (large scale software repository)
- Organized in different (couple hundred) projects: jdt, cdt, emf, ...
- Available via GITHUB
- GIT repositories can be gathered automated via GITHUB’s REST-ful API
- 200 largest Eclipse repositories that actually contained Java code: 6.6 GB Git, 400 MLOC, 250 GB model with 4 billion objects.
Example Plot: Halstead-length for each Revision; Eclipse CDT
Model Create v Analysis Times

![Graph showing the comparison between Model Create and Analysis Times for different components including jdt.ui, xt, eclipselink, jdt.core, swt, cdt, ocl, ptp, org.aspectj, and cdo. The y-axis represents time in hours, and the x-axis represents various actions such as checkout, parse, save, load, merge/increment, and udf. The graph includes box plots and bar charts illustrating the average time per revision in milliseconds.](image-url)
Diskspace

GIT repository vs model size

GIT size
Model size

Diskspace in GB

jdt.core  cdt  jdtui  cdo  emfstore.core  emf  jdt.debug  mf.compare  emf.teko  merge.core
Delta-Compression

Model Creation v Analysis with Delta-Compression

without compression

with compression

- udf
- merge/increment
- load
- save
- parse
- checkout

(time (hours))
Conclusions

- MSR can support software evolution and helps to understand software evolution
- Traversing a source code repository to gather information (MSR) is very time consuming, especially with iterative analysis
- It is possible to save most of this time via saving data in its model state, at the cost of comparably large models that need to be persisted
- The MSR analysis execution time savings are considerable