

# Change-Oriented Repair Propagation

**ICSSP/ICGSE'22**



**Luciano Marchezan\***



Alexander Egyed



Wesley K. G. Assunção



Roland Kretschmer.



**JOHANNES KEPLER  
UNIVERSITÄT LINZ**

Altenberger Straße 69  
4040 Linz, Österreich  
[www.jku.at](http://www.jku.at)

# Overview

1. Illustrative Example
2. Context and Motivation
3. The CORP approach
4. Evaluation
5. Concluding Remarks

# Illustrative Example

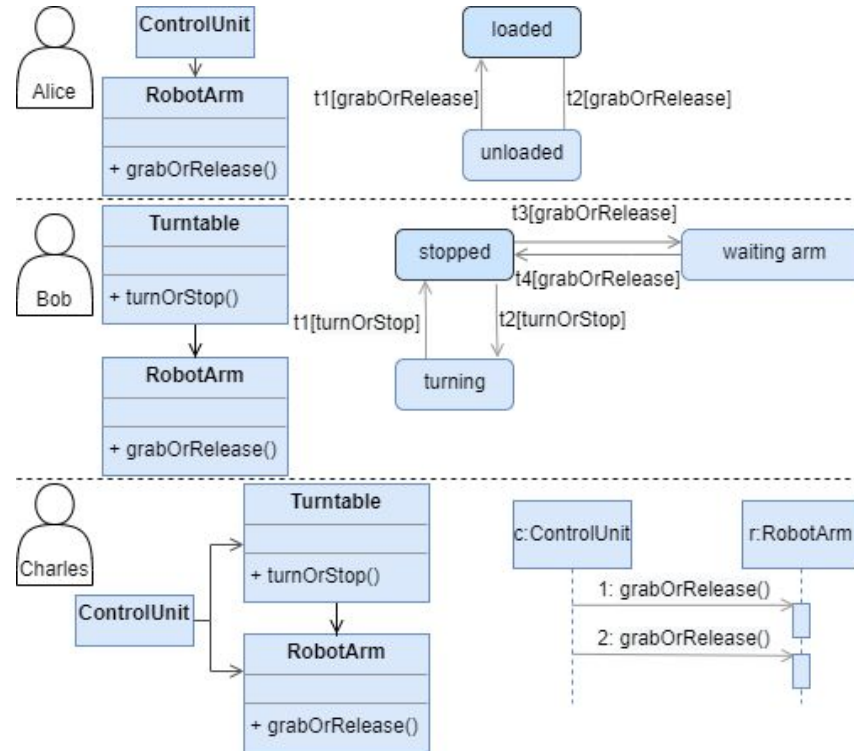


Figure 1: Artifact snippets of three engineers from a robotic arm project

# Illustrative Example

1. Alice splits operation `grabOrRelease()` into two distinct operations.

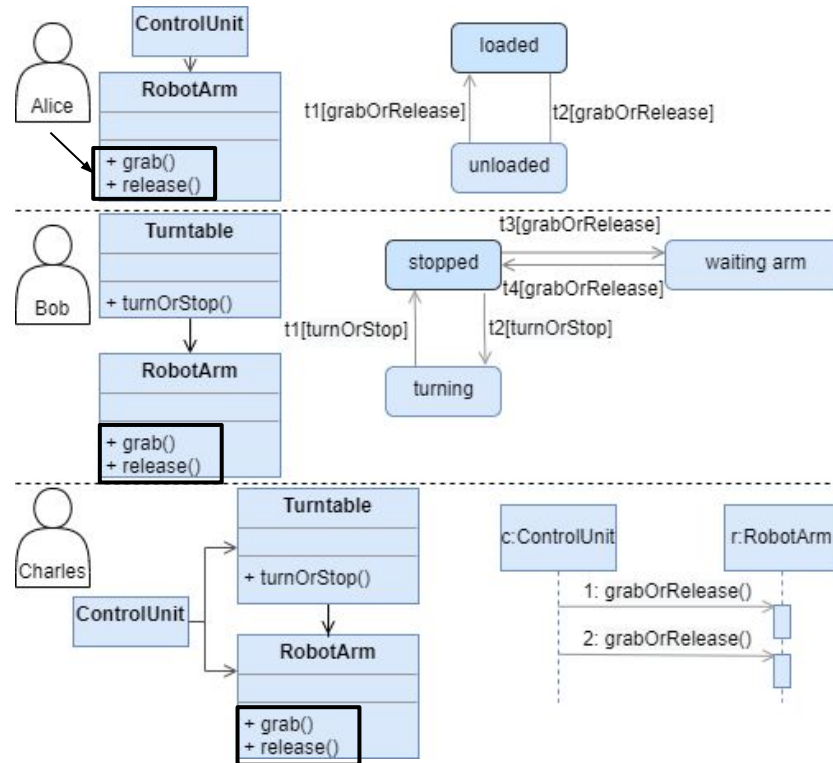


Figure 1: Artifact snippets of three engineers from a robotic arm project

# Illustrative Example

1. Alice splits operation `grabOrRelease()` into two distinct operations.
2. Changes create inconsistencies for all engineers.

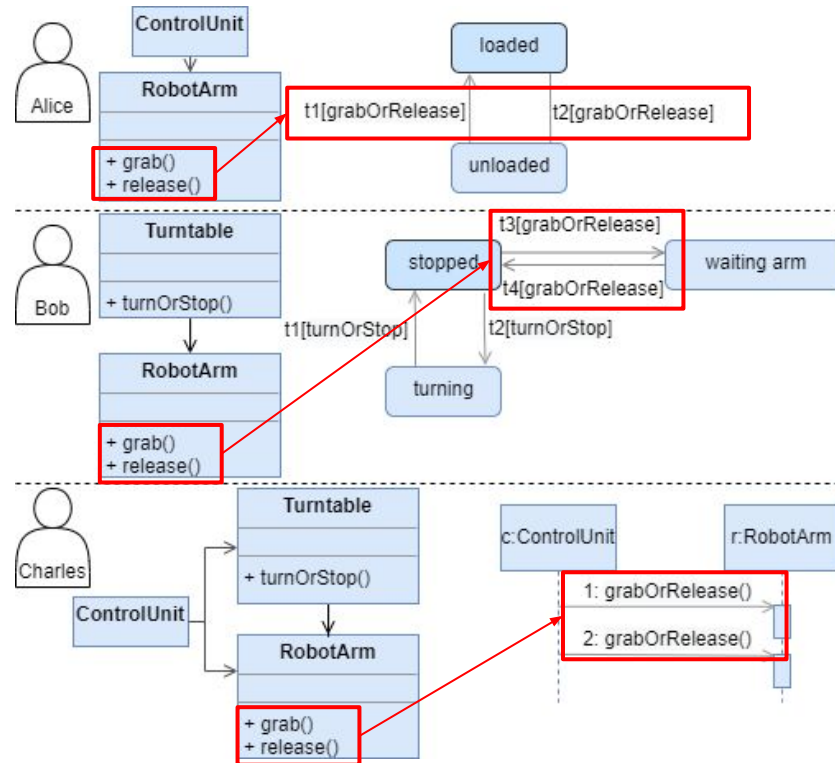


Figure 1: Artifact snippets of three engineers from a robotic arm project

# Illustrative Example

1. Alice splits operation grabOrRelease() into two distinct operations.
2. Changes create inconsistencies for all engineers.
3. Bob applies repairs that conflict with the first changes.

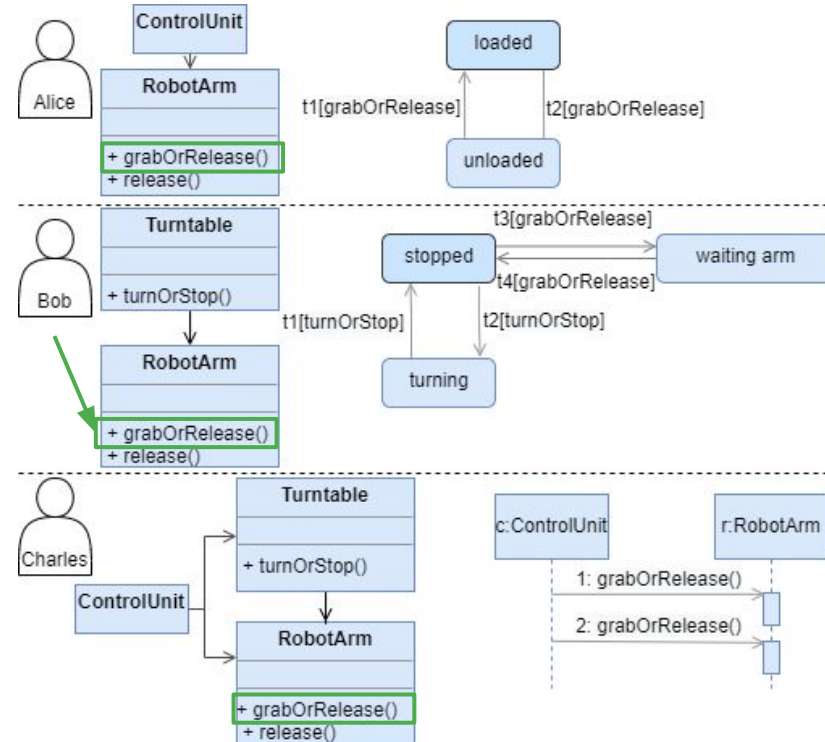
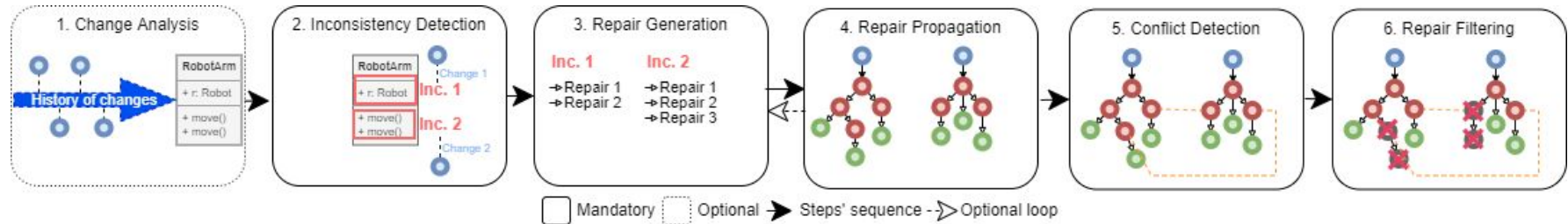


Figure 1: Artifact snippets of three engineers from a robotic arm project

# Context and Motivation

- Impact of artifacts consistency in Model-driven engineering.
- Repairing artifacts is not trivial.
  - Changes applied in one artifact may create new inconsistencies.
  - One repair may conflict with changes from other engineers.
  - Large number of repair alternatives.

# The CORP Approach

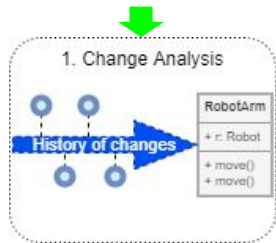


Approach's documentation



# 1. Change Analysis

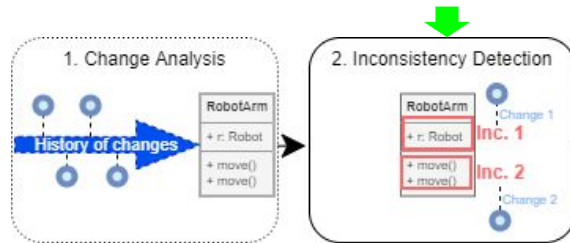
- Analysis of the history of changes is performed in a range of changes defined by the engineer;
- Changes are extracted and stored to be used in next steps;
- This step is not mandatory for the approach to work.



☐ Mandatory ☐ Optional ➔ Steps' sequence -> Optional loop

## 2. Inconsistency Detection

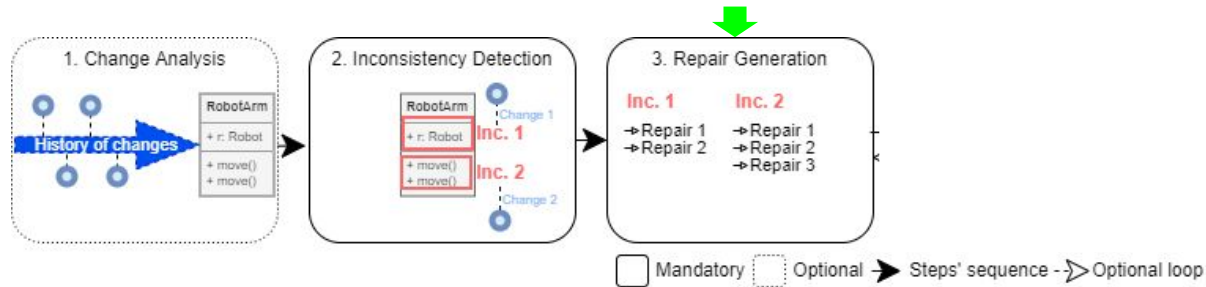
- Based on Consistency Rules defined in OCL;
- Can be triggered in two different ways:
  - Dynamically, *i.e.*, whenever a change is performed in an artifact;
  - Statically, *i.e.*, at any given moment it may be triggered by an engineer;
- An inconsistency can be linked to a change from the history (the cause);



☐ Mandatory ☐ Optional → Steps' sequence -> Optional loop

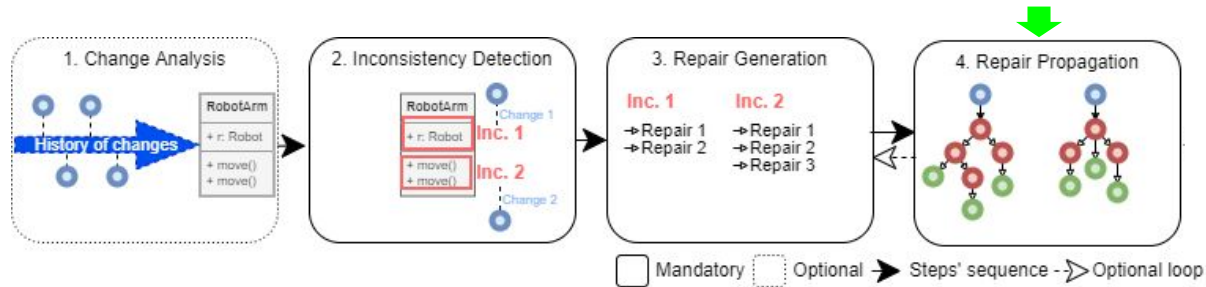
# 3. Repair Generation

- For each inconsistency identified, a set of repairs is generated;
- Repair alternatives can be generated based on the information extracted from the history of changes:
  - e.g., Alice initial change (rename operation *grabOrRelease* into *grab*), could be used to generate a repair (rename message *grabOrRelease* into *grab*).



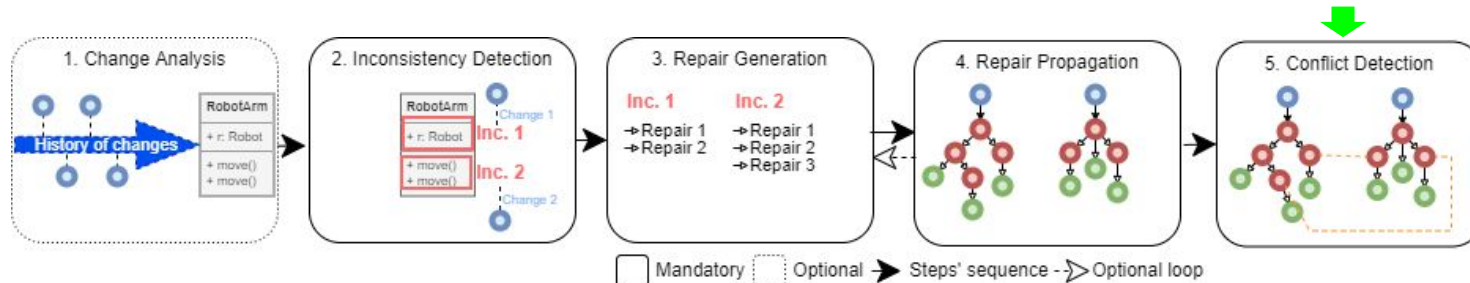
# 4. Repair Propagation

- A copy of the model is created (to preserve the original);
- Repair alternatives are performed in this copy (simulation);
- After a repair is executed, side effects are analyzed (e.g., new inconsistencies);
  - If new inconsistencies were created, then new repairs are generated (Step 3)
- Propagation ends if: There are no new inconsistencies **OR** Repairs can no longer be executed;



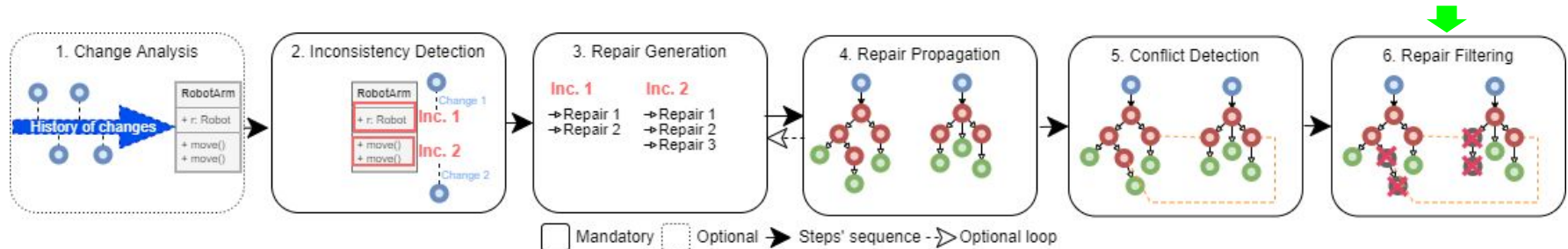
# 5. Conflict Detection

- Checks if repairs conflict with each other (e.g., undo each other);
- Checks if repairs undo changes from the history of changes;
- Creates a link between repairs/changes that have conflicts.



# 6. Repair Filtering

- Engineers may remove conflicting repairs from the available options;
- The engineer must select a path (from initial state until an end state);
- The approach executes the selected repairs into the original model.



# Evaluation

- RQ1. Can CORP propagate repairs to fix an inconsistency, considering all repair alternatives and their side effects?
- RQ2. What is the runtime required for propagating repairs and detecting conflicts among them?
- RQ3. Can the filtering mechanism be used for filtering out unwanted repairs?



# Evaluation Procedure

- We performed an evaluation using a prototype implementation of the approach.
- Dataset is composed of 11 UML models and 14 UML consistency rules.
- Each UML model has at least 3 different versions.
- We extracted changes from the model versions to:
  - generate concrete repairs;
  - identify conflicts with the repairs generated;
  - check how the inconsistencies found in version 1 were fixed.





# Evaluation Data Set

Table 1: UML Models used in the Evaluation

| Name     | Size | Versions | Total Changes | Inconsistencies |
|----------|------|----------|---------------|-----------------|
| GCS      | 203  | 6        | 5453          | 11              |
| SPEM     | 237  | 9        | 91            | 10              |
| T.Agency | 275  | 6        | 2428          | 18              |
| TIM      | 291  | 6        | 2896          | 18              |
| Ball     | 393  | 4        | 2474          | 6               |
| Auction  | 825  | 10       | 7490          | 84              |
| Glide    | 956  | 3        | 923           | 89              |
| JCGrid   | 3470 | 7        | 4927          | 500             |
| Globus   | 4115 | 4        | 4642          | 364             |
| DSpace   | 8856 | 4        | 9550          | 722             |
| ODDT     | 9868 | 6        | 6014          | 1094            |



# Evaluation Results

- RQ1: CORP was able to propagate between 2.31 and 3,511.09 repairs on average per model, always reaching at least 1 consistent state.
- RQ2: Average runtime per model (ms):
  - Repair propagation: min = 5.78, max = 1415.65.
  - Conflict detection: min = 1.1 max = 3748.63.
- RQ3: The number of conflicts ranged from 0.31 to 1735.09 on average per model.
  - The number of conflicts is directly impacted by the number of repair alternatives generated.



# Conclusion - Contributions

- i. A repair propagation approach that can use the history of changes for repair generation;
- ii. A conflict detection strategy that can be used for filtering repairs;
- iii. A data-set with 11 UML models with different versions that can be used for extracting real changes and comparing different model versions;
- iv. An empirical evaluation evidencing results of our approach applicability in the models analyzed within a reasonable amount of time.

# Conclusion - Future Work

- Investigate strategies to solve conflicts among repairs (e.g., merge repairs);
- Perform an user study to evaluate the usability of the approach;
- Explore the applicability of the approach in different contexts (e.g., software modernization)

# Thank you!



Q&A

Luciano Marchezan

[luciano.marchezan\\_de\\_paula@jku.at](mailto:luciano.marchezan_de_paula@jku.at)

<https://isse.jku.at/>



Approach's  
documentation



Repository for the  
evaluation data

**JOHANNES KEPLER  
UNIVERSITÄT LINZ**

Altenberger Straße 69  
4040 Linz, Österreich  
[www.jku.at](http://www.jku.at)