Impact Analysis - Change Propagation

Idaho State University | Computer Science

Isaac Griffith

CS 4423 and CS 5523
Department of Computer Science
Idaho State University

ROAR

Change Propagation

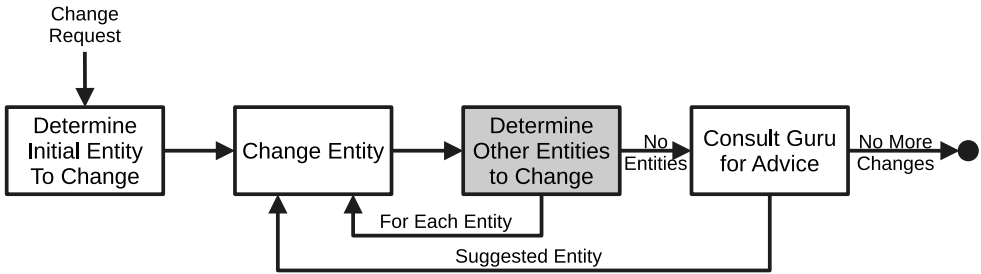
CS 4423/5523

ROAR

Idaho State University
Computer Science

Change Propagation Model

- Change propagation means that if an entity, say, a function, is changed, then all related entities in the system are accordingly changed.
- Based on the work of Hassan and Holt, a change propagation model has been illustrated in Figure.



ROAR

Recall and Precision

- Gurus rarely exist and comprehensive test suites are generally incomplete in large maintenance projects.
- Therefore, software maintenance engineers need good change propagation heuristics, that is, good software tools that can guide them in identifying entities to propagate a change.
- The heuristic should possess the high **precision** attribute to be accurate and the high **recall** attribute to be complete.
- We explained the concepts of **recall** and **precision** before.
- Next, we explain the use of those two metrics to measure the change propagation heuristic by means of an example.

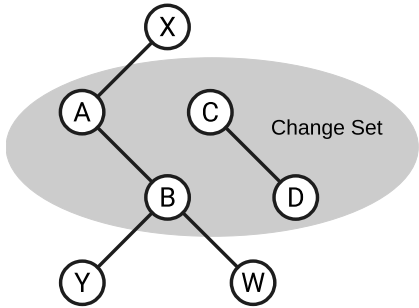
Example

- Let us assume that Rohan wants to enhance an existing feature of a legacy information system.
  1. He identifies that entity A should be changed
  2. After changing A, a heuristic tool is queried, which suggests entities B and X
  3. B is changed and he determines X should not be changed
  4. The tool is informed that B was changed, it then suggests to change Y and W
  5. Neither Y nor W need to be changed and they are left alone
  6. Rohan consults a Guru, Krushna, who indicates C should be changed
  7. C is modified and the heuristic tool is consulted
  8. D is suggested by the tool
  9. D is changed and Krushna is further queried.
  10. Krushna does not suggest any more entities for change
  11. Rohan stops changing the legacy system.

Recall and Precision

- The example entities and their interrelationships are shown in Figure

- The set of entities that are changed will be called change set ; change = {A, B, C, D}.
- The set of entities suggested by the tool is called a predicted set. In the Rohan example, predicted = {B, X, Y, W, D}.
- The entities that were required to be predicted, but were found from Guru, are put in a set call the occurred set.



- In the Rohan example, occurred = {B, C, D}. The occurred set does not include A, which was initially selected by Rohan, because there is no need to predict it. That is, occurred = change - {initial entity}.

Recall and Precision

- Now, recall and precision for this example are computed as follows:

$$recall = \frac{|predicted \cap occurred|}{|occurred|} = \frac{2}{3} = 66\%$$

$$precision = \frac{|predicted \cap occurred|}{|predicted|} = \frac{2}{5} = 40\%$$

Recall and Precision

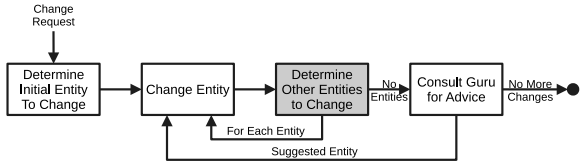
In the analysis of the example is to measure recall and precision the authors, Hassan and Holt, made three assumptions:

1. **Symmetric suggestions:** This assumption means that if the tool suggests entity F to be modified when it is told that entity E was changed, the tool will suggest entity E to be modified when it is told that entity F was changed. This assumption has been depicted in Figure by means of undirected edges.
2. **Single entity suggestions:** This assumption means that each prediction by a heuristic tool is performed by considering a single entity known to be in the change set, rather than multiple entities in the change set.
3. **Query the tool first:** This assumption means that the maintainer (e.g., Rohan) will query the heuristic before doing so with the Guru (e.g. Krushna).

Idaho State University
Computer Science

Change Propagation Heuristics

- The “Determine Other Entities to Change” step is executed by means of several heuristics.
- The set of entities that need to be changed as a result of a changed entity is computed in the aforementioned step.
- Modification records are central to the design of the heuristics.
- In general, source control repositories are used to keep track of all the changes made to files in the system.
- Each heuristic discussed here is characterized by:
  1. data source.
  2. pruning technique.



ROAR

Heuristic Information Sources

- The objectives of the heuristics are to:
  1. ensure that the entities that need to be modified are predicted.
  2. minimize the number of predicted entities that are not going to be modified.
- Some potential information sources are as follows:
  - Entity information
  - Developer information (DEV)
  - Process information
  - Textual information

Entity Information

In an heuristic based on entity information, a change propagates to other entities as follows:

- If two entities changed together, then the two are called a historical co-change (HIS).
- Static dependencies between two entities may occur via what is called CUD relations: **call**, **use** and **define**. A **call** relation means one function calls another function; a **use** relation means a variable is used by a function; and a **define** relation means a variable is defined in a function or it appears as a parameter in the function.
- The locations of entities with respect to subsystems, files, and classes in the source code are represented by means of a code layout (FIL) relation. Subsystems, files, and classes indicate relations between entities – and, generally, related entities simultaneously.

Additional Heuristics

Developer information (DEV)

- In an heuristic based on developer information, a change propagates to other entities changed by the same developer.
- In general, programmers develop skills in specific subject matters of the system and it is more likely that they modify entities within their field of expertise.

Process information

- In an heuristic based on process information, change propagation depends on the development process followed.
- A modification to a specific entity generally causes modifications to other recently or frequently changed entities.

Additional Heuristics

Textual information

- In an heuristic based on name similarity, changes are propagated to entities with similar names.
- Naming similarities indicate that there are similarities in the role of the entities.

Pruning Techniques

- A heuristic may suggest a large number of entities to be changed. Several techniques can be applied to reduce the size of the suggested set, and those are called pruning techniques.
  - **Frequency** techniques identify the frequently changing, related components. The number of entities returned by these techniques are constrained by a threshold.
  - **Recency** techniques identify entities that were recently changed, thereby supporting the intuition that modifications generally focus on related code and functionality in a particular time frame.
  - **Random** techniques randomly choose a set of entities, up to a threshold. In the absence of no frequency or recency data, one may use this technique.

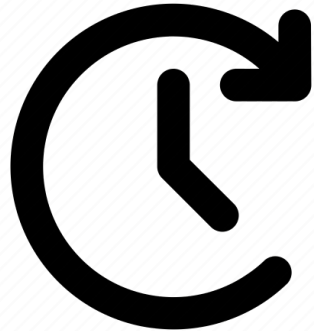
Empirical Studies

- Hassan and Holt studied the performance of four heuristics: **DEV**, **HIS**, **CUD**, and **FIL** using the development replay (DR) framework
- To evaluate an heuristic:
  1. historical changes are re-played
  2. the recall and precision attributes are measured for every change set.
- They used five open software systems – **NetBSD**, **FreeBSD**, **OpenBSD**, **Postgres** and **GCC** – and the performance results are given in Table 6.5.

Application Name	Application Type	DEV		HIS		CUD		FIL	
		Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision
NetBSD	OS	0.74	0.01	0.87	0.06	0.37	0.02	0.79	0.16
FreeBSD	OS	0.68	0.02	0.87	0.06	0.40	0.02	0.82	0.11
OpenBSD	OS	0.71	0.02	0.82	0.08	0.38	0.01	0.80	0.14
Postgres	DBMS	0.78	0.01	0.86	0.05	0.47	0.02	0.77	0.12
GCC	C/C++ Compiler	0.79	0.01	0.94	0.03	0.46	0.02	0.96	0.06
Average		0.74	0.01	0.87	0.06	0.42	0.02	0.83	0.12

For Next Time

- Review EVO Chapter 6.4 - 6.6
- Read EVO Chapter 7.1 - 7.3
- Watch Lecture 17



Are there any questions?

ROAR