Rajlich, V. (2011). **Concepts and Concept Location**. In *Software engineering: The current practice* (pp. 87-104). CRC Press.

Reviewed by Roberto A. Bittencourt
Assistant Teaching Professor

University of Victoria

Summary

Main Goal

- Introduce the main ideas on concepts and concept location and describe practical techniques to perform concept location on source code.

Problem

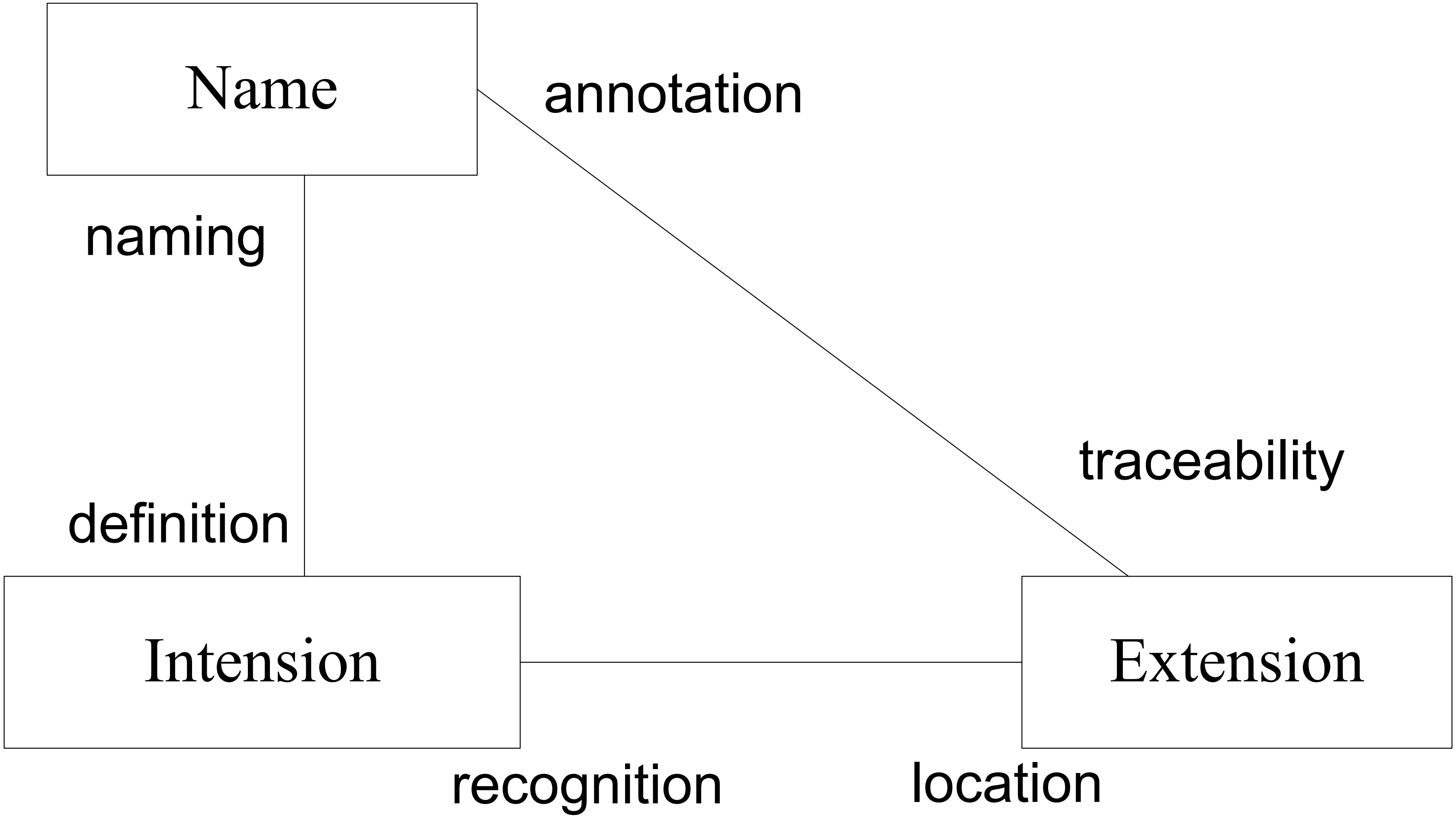
- Finding what source code has to be changed when given a change request is a challenging task in large software systems.

Solution, in short

- The author introduces the basic ideas of concepts and concept location, and describes two main techniques to perform concept location in source code: GREP and dependency search.

Detailed Solution

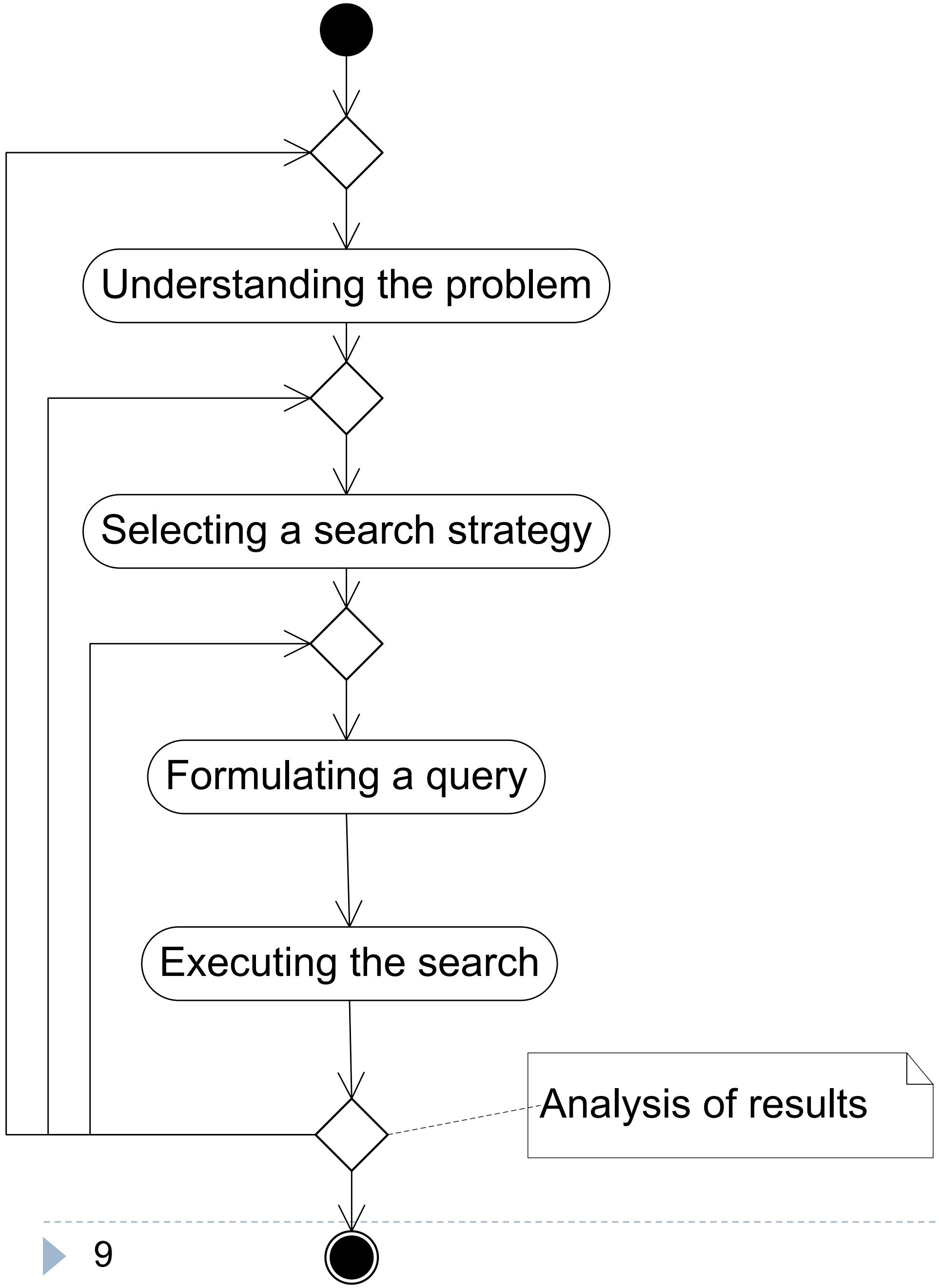
Concept triangle



Concept location

- Concept extensions are implemented as code fragments
  - variables, classes, methods, or other
- Programmers find these code fragments
  - easy in small programs or in the programs that the programmer knows well
  - hard in large programs or programs that the programmer does not know

Search in the unknown parts of system



Concept location techniques

- Human knowledge
- Traceability tools
- Dynamic search (execution traces)
- Static search
  - dependency search
  - "grep" (pattern matching)
  - information retrieval techniques

GREP Search Technique

- GREP is an acronym for "global regular expression print"
  - GREP prints out the lines that contain a match for a regular expression.

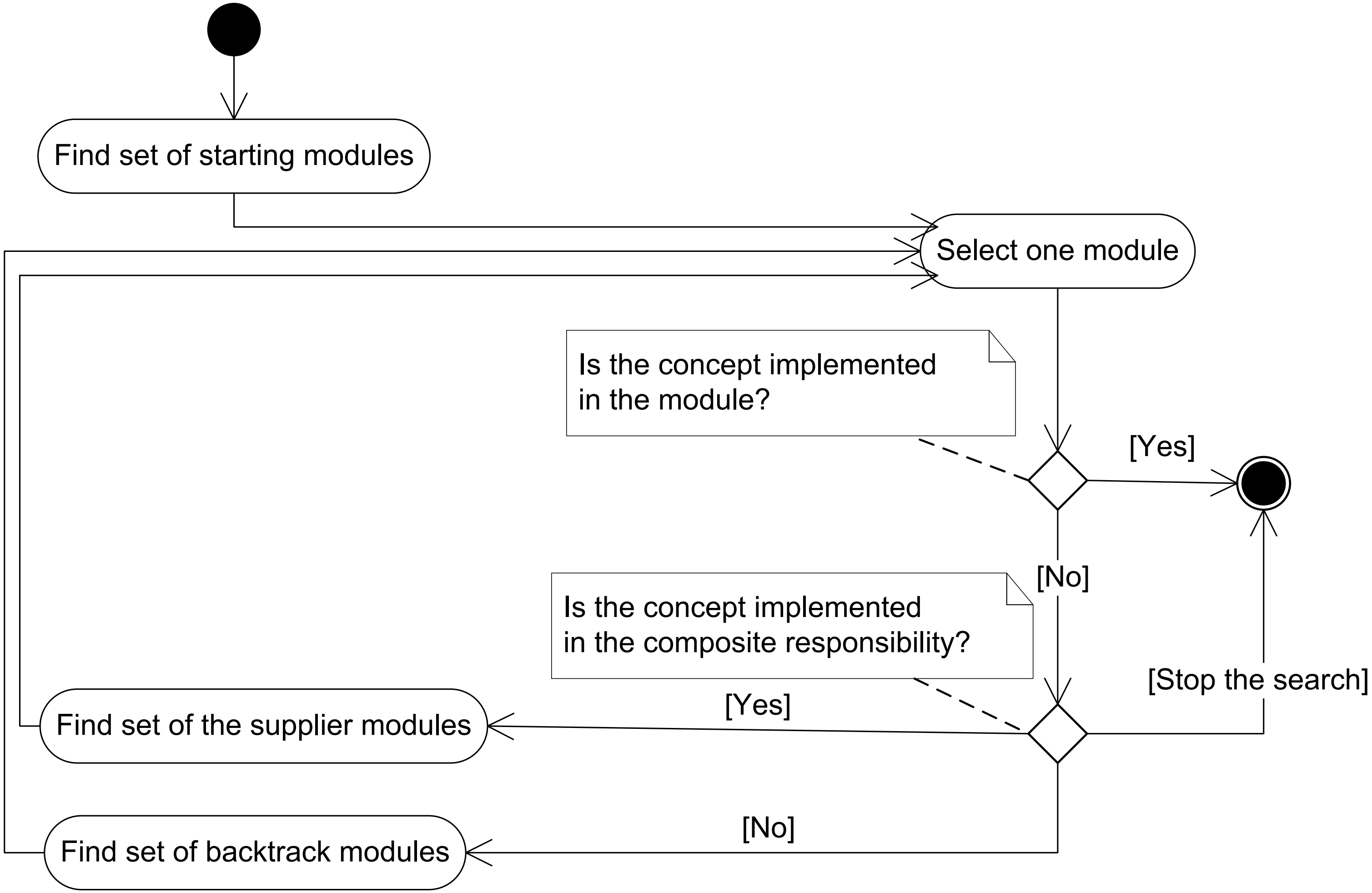
Concept location by GREP

- Classical technique for concept location
  - based on pattern matching
- Programmer formulates a query
  - based on concept name(s)
- grep searches the files
  - finds corresponding lines of code (“hits”)
  - programmer investigates the hits
- If a search fails, new query is tried
  - programmer learns from failed search

Dependency Search Technique

- Uses Class Dependency Graphs (CDG)
  - extracted form the existing code
- Local functionality
  - consists of concepts that are actually implemented in the module and are not delegated to others.
- Composite functionality
  - as the complete functionality of a module combined with all its supporting modules.
- Determined by reading code and documentation

Concept location by dependency search


Find set of starting modules
Select one module
Is the concept implemented
in the module?
[Yes]
[No]
Is the concept implemented
in the composite responsibility?
[Stop the search]
[Yes]
Find set of the supplier modules
[No]
Find set of backtrack modules

Critical Evaluation

Positioning on the text

- In general, I agree with the author about the challenges described in concept location, and also with the main solutions to concept location based on GREP and dependency search.
- I do not agree much with some of the author's ideas on interactive tools to perform concept location. I believe developers usually prefer to perform concept location on their own by means of exploring the software componentes (files) via an IDE.

Pros

- Clear description of the idea of software concept as well as the task of concept location
- Clear description of concept location techniques, helped by the use of examples
- Description of the challenges developers face when performing concept location

Cons

- Concept location based on dependency search is useful in monolanguage software systems; when systems use different languages and frameworks, it is harder to perform dependency-based concept location
- Concept location is intimately related to software behavior, which is hard to get from a structural point of view only; solutions to concept location may also exercise code runs by means of unit, integration and system testing

So what?

- I liked very much the author's introduction to concepts and concept location; readers may easily grasp the basics on the topic
- I also like that the author shows practical techniques to perform concept location by using tools from an IDE
- I have also learned that the challenges of concept location are essential issues of software engineering, since stakeholders will usually describe software in terms of features/concepts, while source code is usually organized in a very different way (e.g., components, packages, files, classes, methods)

Rajlich, V. (2011). **Concepts and Concept Location**. In *Software engineering: The current practice* (pp. 87-104). CRC Press.

Reviewed by Roberto A. Bittencourt
Assistant Teaching Professor

University of Victoria