Concern Based Comprehension

CPRE 416-Software Evolution
and Maintenance-Lecture 10

Contents

- Review of the formal definition of *concern* as given in the Robillard Ph.D. thesis.
- Going beyond Robillard's definition.
  - Formalism vs. practice
  - Generalization of the concept of concern.
- A real-world application.

Subsetting Mechanism

- Relations provide a mechanism to form interesting subsets.
- Given a set *S*, an element *x*, and a relation *R*, the subset R_X is defined as:
  - {*Y* | *X R Y*, i.e. all elements related to X}.
- If R is an equivalence relation then the subsets actually form a partition of the given set, i.e. distinct subsets are actually disjoint.

Why Fragments

- Fragments is nothing but a subsetting mechanism. It divides program artifacts into *interesting subsets*.

Formalism vs. Practice

- Formalisms are interesting if we can apply them to solve real-world problems.
- Converting an useful formalism or principle into practice can be very challenging.
- We will now follow up with concrete examples.

From Goals to Concerns

- Among the three types of reading code, step-by-step reading with vague goals is often not practical with large code.
- First, we must set some goals so that our understanding process will have well defined and useful outcomes – easier said than done.
  - Let us discuss some goals for Homework 3 and the possible outcomes.
- Next, think of transforming goals into specific concerns. If done successfully, it will take us from an abstract problem to concrete steps that we can perform with a real code.

Examples

- Next we will consider some specific examples of goals.
- We will convert those goals into concerns.

Following Operation Sequences

- A common comprehension need is to understand the code to see if certain operations are sequenced properly.
- Examples:
  - An open operation is performed before writing to a file.
  - A shared data structure is locked before accessing it.
  - A memory allocation is followed by deallocation.

Understanding Subsystems

- Often, a given task requires you to understand a subsystem.
- Examples:
  - Understand and modify the file subsystem of an OS. (HW3).
  - Understand and modify the subsystem that handles adds and drops in a course registration system

Goals to Concerns

- Next, we will see how we can formulate concerns to address our comprehension goals.
- Suppose we find that a file is written in a function called update_record(), but that function does not open the file.
- We need to check callers of the function to see if the file is open by a caller.

Formulation of Concerns

- We will see that the process of checking if the open operation is performed by some caller is done systematically by formulating an appropriate concern, i.e. by defining the so called fragments.
- Let R_X represent the relation *Calls*
- Consider the fragment *S1*:
  - *S1* = {*f* | is a method such that *f* calls *update_record()* }
  - Alternatively, *S1* = { *f* | (f, *update_record()*) *belongs to* R_X}

Formulation of Concerns

- The concern *S1* is useful but not enough to address our goal. Why?
- We need to apply the *Calls* relation transitively so that we can get not just the immediate callers of *update_record()* but also their callers and their callers and so on.
- The fragment *S2* does the job:
  - *S2 = TC((calls, update_record()))*
- Note that the S2 can be shown as a graph and it is called the reverse call order graph.

Formulation of Concerns

- Next we need to identify the elements of S2 that actually open the file. This is again a subsetting operation and can be represented as a concern. How?