Workflows

Steven Zeil

September 4, 2013

Contents

Workflows

Certain activities seem to occur almost continuously. The *workflows* describe the “daily work” of a software designer. The four workflows are

- Analysis
- Design
- Implementation
- Validation

. . . . . .

1 What Happens In Each Workflow?

The Topic of Interest

The *topic of interest* (TOI) is the part of the system that you are working with at the moment.

- might be the entire system or a single ADT or even a single function
- depending upon where you are in the overall process

All the workflows have to be viewed in terms of the TOI.

. . . . . .

All the workflows have to be viewed in terms of a part of the system that you are working with at the moment. That "part" might be the entire system or a single ADT or even a single function, depending upon where you are in the overall process. (More on that idea shortly when we look at software development process models.) We’ll call that part of the system the *topic of interest* (TOI). The TOI could be the entire system or a single ADT or even a single function within an ADT. For example, in the early stages of design you are concerned with the entire system, so the TOI is the entire system. As you begin to decompose the system into smaller parts, you begin to focus your attention on those parts one at a time. Each time you do this, your TOI becomes progressively more narrow. On the other hand, when you are testing the system, you may start with unit testing where your TOI is the development of tests for a single ADT. Later your focus will broaden, as during integration testing your TOI will be a subsystem, and during system testing the TOI expands to comprise (testing of) the entire system.

Whatever the TOI is, at any moment in time, software developers can expect to engage in the following 4 generic activities:

What Happens in Each Workflow?

- *Analysis* seeks to describe *what* is (or should be) going on in the topic of interest.

  In analysis, we consider the "world" surrounding the TOI and try to learn how the TOI must interact with that world.

  The focus of analysis is external, and there is an emphasis on interfaces.

. . . . . .

What Happens in Each Workflow?

- *Design* considers the questions of *how* our TOI will accomplish the goals established in the most recent round of analysis.

  The focus of design spans the external and the internal.

- *Implementation* is the rendering of the design decisions into a final form.

  "Final" in this case refers only to this round of work on the TOI.

- *Validation* consists of any activities we employ to determine whether the implementation is acceptable and whether it is time to move on to a different TOI.

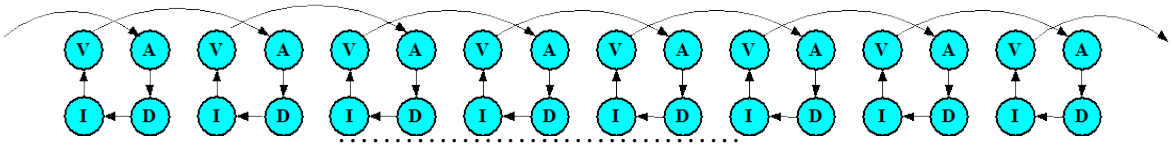
  Validation can take many forms, including testing, reviews with other team members or with the domain experts, and checks for completeness and internal consistency.

. . . . . .

2 Iterative Application of the Workflows

Iterative Application of the Workflows

- At the low level, a continuous cycle through the four workflows
- The TOI changes, but the daily activities represented by the 4 workflows remains the same.



At the low level, we should consider the total process of developing a software system as a continuous cycle through the four workflows, analysis, design, implementation, and validation. The TOI changes, but the daily activities represented by the 4 workflows remains the same.

3 Examples

Examples

A few examples of a possible single "turn" through the ADIV cycle follow.

. . . . . .

3.1 Example: Class Discovery

Example: Class Discovery

Topic of interest:

What are the basic characteristics of a book in the library?

. . . . . .

Analysis:

- Many of the characteristics are quite obvious. Books have titles, authors (possibly multiple), publishers, ISBN numbers, etc.
- However, someone looking through the library's card catalog notes that many of the books are labeled "copy 1", "copy 2", etc.
  - Based on this observation, the team got into a discussion of whether the term "book" actually refers to the thing on the shelf or to something more abstract. Is "War and Peace" a book even if the library does not own a copy? If the library has two copies, what do we call the common "thing" that these are copies of?

. . . . . .

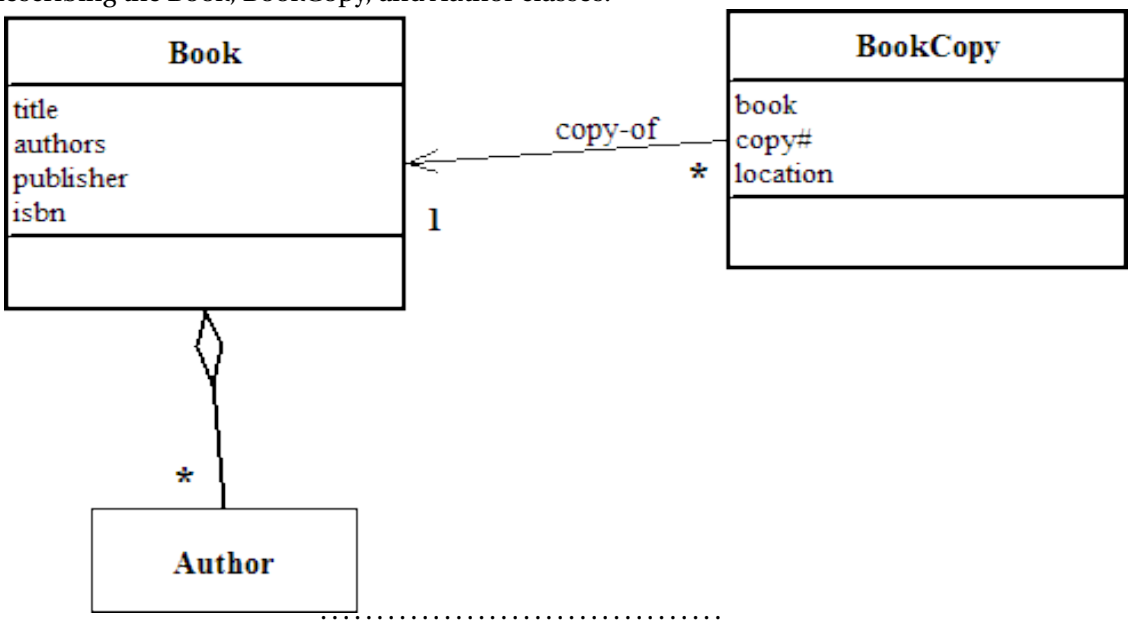
Design:

- The team proposed that the term "book" be used for the abstract idea of the thing that an author writes.
- The term "copy" or perhaps "bookcopy" will be used for a physical copy of a book.
- Books have all the attributes previously described.
- Book copies have attributes of location, copy number, and, of course, the book that they are a copy of.

. . . . . .

Implementation:

The team draws a simple UML class relationship diagram (these diagrams are introduced in a later lesson) describing the Book, BookCopy, and Author classes:



Validation:

- The team discusses with a librarian their decision about proper terminology and about the key attributes.
- The librarian suggests that the distinction between "book" and "copy of a book" is technically correct, but often muddled in common use.
- The librarian notes that books have other important attributes, including date of publication and subject area. Finally, the librarian suggests that the focus on books may be too narrow, as the library actually houses many different kinds of publications.

3.2 Example: ADT Interface

Example: ADT Interface

Topic of interest: Developing the ADT interface for a `Book`.

Analysis:

The developer looks at the above diagram and notes that each book can have multiple authors, so the interface must provide some way to access an unknown number of author objects.

. . . . . .

Design:

The developer suggests using an iterator for this purpose, knowing this to be a common idiom in C++ for accessing an unknown number of "contained" objects.

. . . . . .

Implementation:

The developer adds the lines

```
typedef ... AuthorIterator;
```

and

```
AuthorIterator begin() const;
AuthorIterator end() const;
```

to the class declaration for `Book`.

. . . . . .

Validation:

The developer looks at some of the proposed algorithms that will work with books and authors, checking to see if those algorithms could be coded using an iterator style.

Satisfied that it will do so, the developer distributes the new class declaration to the rest of the development team for their approval.

. . . . . .

3.3 Example: ADT Implementation

Example: ADT Implementation

Topic of interest: Implementing the Book ADT

. . . . . .

Analysis:

- The programmer reviews the required attributes and behaviors of the book class and concludes that it is pretty straightforward, except for the question of how to store the authors.
- Reviewing the various application algorithms that manipulate books and authors, the programmer notes that adding and removing authors is fairly uncommon.

– The programmer also notes that there are no instances of needing "random access" to the authors. The only algorithms that access the authors will process each author in turn.

. . . . . .

Design:

The programmer decides that the `std::list` would provide appropriate performance and would make for a simple implementation as well.

. . . . . .

Implementation:

The programmer writes the std list into the private section of the class declaration and codes the accompanying function bodies.

. . . . . .

Validation:

The programmer runs the previously developed self-checking unit test on the new `Book` implementation.

. . . . . .

3.4 Example: Training Manuals

Example: Training Manuals

Topic of interest: Preparing training documents for the new automated check-in system

. . . . . .

Analysis:

- The author reviews the (use-cases) scenarios for interactions between library staff, patrons, and the other library objects relating to check-in (return) of copies.
- The author makes a list of those that will require explanation to the library staff.

. . . . . .

Design:

The author decides which scenario variants and interactions are important and common enough to be needed in the initial training, and which can be relegated to reference documentation.

- The common and important items are arranged into related groups, and the author creates an overall outline that encompasses those groups as chapters/sections of the training manual.

. . . . . .

Implementation:

The author writes the relevant chapters and sections.

. . . . . .

Validation:

The sections are submitted to one of the system designers to be checked for accuracy and to one or more domain experts to be checked for understandability.

. . . . . .