Intentional Software

Duong Pham
10/28/2008
Software Engineering
Charles Simonyi is a former executive of Microsoft Corp. and the creator of Microsoft Word, which has been an invaluable tool for computer users for years. From 1981-2002, Simonyi "was employed by Microsoft Corp. where he held titles of Director of Application Development, Chief Architect, and Distinguished Engineer" [2]. He now heads his own company, Intentional Software, through which he introduces a new methodology of writing software. His main idea is to get the most contributions and information out of the domain experts, that is, the customers, in the writing process. By preserving the intentions of the domain experts, the Intentional Software Corporation is introducing a new way to simplify and enhance the productivity of the software production process.

Nowadays, a wide variety of problems can be solved using computer software. Well-written software should be able to solve the problems that are specified in its domain. The creation of such software can be the result of the combination of well-stated problems from the domain experts and skilled experienced programmers. Most of the time, contributions of the domain experts are more significant than those of the programmers. Although domain experts are the main component, their instructions are not understood and executable by computers and need to be converted by the programmers. From this point of view, maximizing the range of participation of the domain experts will greatly increase the sufficiency and accuracy of the software creation process.

The traditional method of programming is direct programming, where the problems are stated by the domain experts, and the programmers turn them to the source code that can run on the computer. This direct development work-flow is popular and suitable for creating small and simple software. In the larger scale, there are persistent problems with this process. Those problems arise from the fact that maintenance and correction need to be stated and implemented separately. This is difficult to achieve in a complex system since "a more complex domain vocabulary, domain relationships, and domain rules simply cannot be mapped into a programming language without a great deal of software engineering trickery" [1].
Now it is time for direct programming to be replaced by a better method, generative programming. To have a high level of intentionality, the problems and their solutions should be expressed in the domain terms. This is difficult to do because the expressions are not executable by the computer. The solution for this is to use the technique where the programmers focus on transforming the domain code into the executable code by writing a generator program. Once the generator program is written, the domain experts will distribute changes to the final program directly and accurately. The intentions of the domain experts are preserved and utilized in the software creation process.

Intentional Software comes in as a promising alternative that gives domain experts the flexibility to record and maintain their contributions. The Intentional Software consists of three parts: domain schema, domain code, and generator code [1]. The domain schema serves as the dictionary which will be referenced frequently by the generator. The domain code is written by the domain experts. The programmers will focus on writing the generator which transforms the domain code into the target code in some specified programming language. This target code can be executed by the computer and therefore is the final program. The most significant part will be writing the domain code because it is where the whole program will be generated from.

The benefit of using the Intentional Software is the separation of the domain experts and the programmers. They are working on different parts of the process of creating the final program. Domain experts can utilize their skills to express the problems in a chosen language and programmers can concentrate on what they do best, which is writing executable code. This separation theoretically speeds up the software creation process and makes maintenance and correctness easier.

Questions rise for the following issues: “What is the language of the domain code? Who defines and who supports this language of the domain code? What if the problem spans more than one domain? And what API does the generator use to access the domain code”?[1] The resolution to the existing issues
and an aid to creating Intentional Software is to use the Domain Workbench, created by the Intentional Software Company, for support.

The process of creating Intentional Software begins with writing the domain schema. Supported by the programmers, the domain experts will define the domain schema as the interface between the Domain Workbench and the generator as well as between the domain users and programmers. The domain schema should contain only the terms that will be referenced by the generators. Each concept of the domain will be represented by a distinct identity. Next, using the domain schema, the domain experts write and maintain the domain code. A specific programming language is no longer needed due to "the Domain Workbench's ability to project contents from multiple interrelated domains in multiple views" [1]. Finally, the programmers, using their implementation knowledge, write the generator to transform the domain code to the intended implementation. The generator program can access the domain schema and the domain code through an API provided by the Domain Workbench. Multiple domains can be defined, created, edited, transformed, and integrated by the Domain Workbench [1].

An intentional tree is used to display the notations that are used in the program with nodes and branches. The tree is a projection of the domain code. Each node points to an intentional definition in the domain schema. The representation of the domain code helps the domain experts and programmers to visualize the connections among the definitions and how to refer to them when writing the generator. The programmers should be able to redefine or add or remove any node of the tree without disturbing the existing structure. Refinement is used to “add operands, attributes, properties, formatting, subscripting, tags, or decorations” [1]. Trees from different domains can be merged by mixing their domain schema because each node is self-identifying through its schema.

For example, the domain code for a formula may look like this:
return \( a = \frac{b}{c + 1} \);

The intentional tree will project this formula as:

```
Plus
  {  
    Div
    {  
      b,  
      c  
    },  
    1  
  }
```

In domain schemas, domain definitions are defined as nodes and represented in the intentional tree and called “meta-domain” [1]. The description of this meta-domain is not discussed in the original article. Domain code and domain schemas do not have any behaviors to them. Transformations add behaviors to them and one special “type of transformation is the projection for editing in the Domain Workbench” [1]. In this editing model, users can cut, copy, and paste parts of the intentional tree to the others, just like in text-based environment.

The capabilities of the Domain Workbench to define, create, edit, transform and integrate multiple domains give engineers an opportunity to rethink some software engineering practices [1]. The focus of the domain schema divides into two ways of developing software: Language Oriented Development and Domain Oriented Development. Language-oriented development is where the notations, the schemas, and the generator closely mirror and extend existing general purpose languages [1]. In this type of development, it is possible to mix sources from multiple languages, extend a language, eliminate code copying and aspect fragmentation, and reuse the code. Domain-oriented development begins at declaring the problems in business domain and a generator is built to create solutions. The domain experts can make changes in the domain code and redistribute them by running the generator. The programmers can substantially implement the generator to apply changes.
During the code process, programmers are prone to making errors because they are human. In a complex program, applying changes specified by the domain experts is very likely to break working codes and to introduce new errors. Intentional Software applies domain experts’ desired changes directly from their codes through the generator. Because the language is chosen by the domain experts, it is less likely for them to produce mistakes. Using the Domain Workbench, the creation of Intentional Software becomes practical and easier. The idea of capturing the intentions of the domain experts has been fully explained. The use of generative programming transforms the domain code into the executable final program. It introduces a new programming method which is more direct from domain experts. In this case, the domain experts will change the final program directly without breaking the currently working codes and introducing new errors. Finally, productivity of the software production process can be greatly simplified and enhanced by using the intentional software.
Work cited

  

(2) Intentional Software Corporation, 2002-2008
  
  http://www.intentsoft.com/company/management.html