

Merging Multiple Conventional Models in One Stable Model

Stability models may demand greater investment in analysis, but when used wisely, savings in development and maintenance costs more than make up for it.

Unlike the products of other engineering fields, there is no physical deterioration to cause software to fail. Most software defects and deterioration are caused by changes in software [2]. Generally, these changes cannot be avoided. Such changes are often the result of the natural evolution of business processes and changes in the underlying requirements of software systems to meet these evolving needs. To achieve software stability is to balance the seemingly contradictory goals of stability over the software life cycle with the need for adaptability and extensibility.

To accomplish this balancing act, we hypothesize that certain refinements can be applied to conventional OO analysis and design techniques. However, such refinements must not overly complicate our conventional approach. In software, we view simplicity and elegance as synonymous. Simplicity is considered an important characteristic of a good model. To demonstrate the simplicity and elegance of the Software Stability Model (SSM), we apply it to a study of open-pit mining (The detailed description of this problem can be found at ww.cse.unl.edu/~fayad/SoftwareStability/OO_PSLA01-DesignFest-Final1.doc).

The Study

In an open-pit mine, mechanical shovels load material from depots into dump trucks. These trucks then transport material to various destinations. Ore is transported to mineral processing facilities to be processed and prepared for market. Waste is delivered to dumps.

A scheduling or dispatching system is required to assign the empty trucks to their proper destinations. This system checks the status of each truck and shovel. As soon as a truck is free, it is assigned to a shovel that will be free when the truck arrives. If all the shovels are busy, the dispatcher system selects the shovel with minimum wait time.

A conventional model for this problem is given in Figure 1. The model illustrates how the transport system works. Ore is placed into the ore extraction openings, and is then transported to a mineral processing facility. Similarly, waste is placed into waste removal openings and is then transported to waste dumps. Dump trucks are assigned to these openings according to a schedule. Shovels load ore

and waste into these trucks.

The project's business objective is to maximize the mine's profitability and satisfy production quotas. The project will accomplish this by optimizing the equipment, personnel, and the overall efficiency of the mining operation.

We can readily imagine changes in production techniques or mining conditions that will require changes to this software model. Perhaps conveyor belts are used as the main transportation methods instead of trucks. In this method, loaders carry materials from the depot and dump it into feeders. The feeder feeds a crusher at a steady rate and the crusher reduces the size of rocks to a proper size for transporting by conveyor belt. The whole system must be consistent in terms of capacity of materials flow. The loader must handle the required volume of material over time (for example, tons/hour.) The feeder must feed the crusher at the same rate, and the crusher must deliver the same amount of material of the proper size to the conveyor. Conveyors, in turn, must have the proper width and speed to transport the material. If any element of this system fails to maintain the

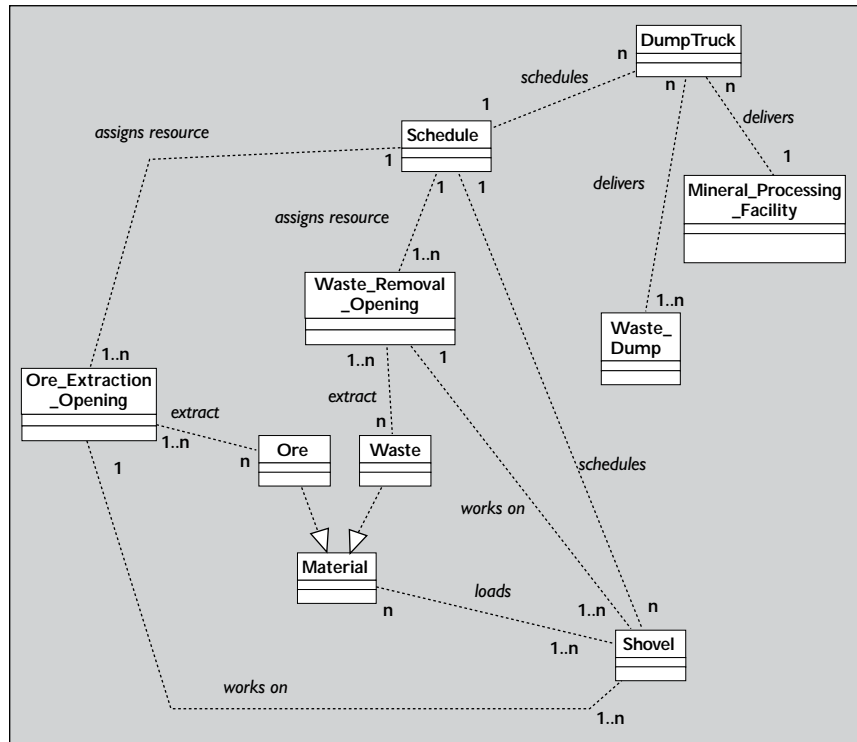
required flow rate, material congestion occurs, and the whole system fails to achieve its goal of transporting the designated quota of material to the destination.

With the introduction of this new technique, the conveyor belt, the old model fails to satisfy the business requirements. We must modify it. The new class diagram might be generated as in Figure 2a. We note that the new model doesn't bear a close resemblance to our original model, nor would it satisfy the requirements of our first open-pit mine.

Other locations may use a pipeline as the transportation mode. With this pipeline technique, material is loaded into feeders by loaders. Each feeder passes the material to a crusher and then on to a ball mill. Ball mills reduce the size of materials and prepare them for transportation through pipelines. The milled material is poured into mixers, where water is added to make slurry. The pipeline then carries the material to the destination.

With conventional models, the tendency is to remodel (as in Figure 2b). Again, the result is a substantial change.

As these examples illustrate, conventional OO modeling is subject to instability. Many changes to the business process most likely to lead to changes in the previous models, prompting a reengineering process. If the previous work is reused, the tendency is to attach new objects to the existing model similar to the way that barnacles



adhere to the hull of a ship.

The Stability Approach

With the stability approach (see Figure 3), we first define the purpose of the system and delineate why the system is needed.

Although this aspect of analysis is not unique to the SSM, the model formalizes the analysis and focuses the analysis activities around the concept of Enduring Business Themes (EBTs). EBTs are the core abstractions of a problem domain. These themes are unlikely to change over time. For example, one important theme of the open-pit mining problem is “efficiency.” The concept of efficiency clearly has a role as an EBT,

Figure 1. Typical class model for open-pit mining.

because efficiency is part of the business objective for building the system. When we derive a schedule for trucks, we are actually trying to make the system work more efficiently, hence, more profitably.

Another important theme is “concurrency.” Consider the interaction between different components of the transport system; they are assigned to work together when they are available. Dump trucks are assigned to shovels when they are available, and they only work together when shovels are available at the same time. In other words, they have concurrency. This concept is also for

Thinking Objectively

material flow through the conveyor belt and pipeline systems.

All of the components in the system have to pass a specific amount of material to the next component within a specified amount of time. All of these material flows have to be equal in a system and all the components of the system must be available and working at the same time to be concurrent. Without this concurrency we cannot imagine a transport system; the result is congestion and failure. Therefore, concurrency is a very important measurement and one of the core purposes of our scheduling system.

After identifying the EBTs associated with our problem, we can identify and associate those business objects (BOs) that provide the necessary abstractions of the processes underscoring the business operations (such as origin, destination, and transport). The third step of the SSM is to define those Industrial Objects (IOs) that refine (or instantiate) the various BOs. In this way, our model exhibits stability over changes. If we substitute feeders, loaders, crushers, and conveyors for shovels and dump trucks, the core model is unchanged. It is through changes to the peripheral objects that we achieve the balance between changes and our goal of stability. Despite the various modifications to support various

instances of the problem, our EBTs and BOs remain stable.

In addition, the model remains well organized over the course of many design iterations. Ultimately, we believe this approach has tremendous benefits for soft-

ware teams. This study shows the SSM is elegant and extensible.

The layered approach of the

Figure 2a. Class diagram of conventional model for conveyor belt transportation.

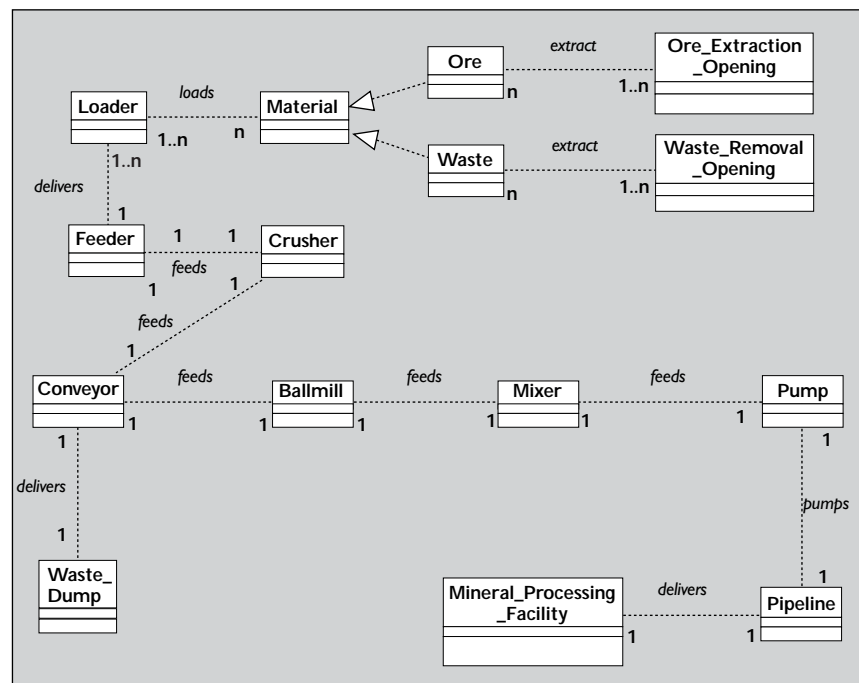
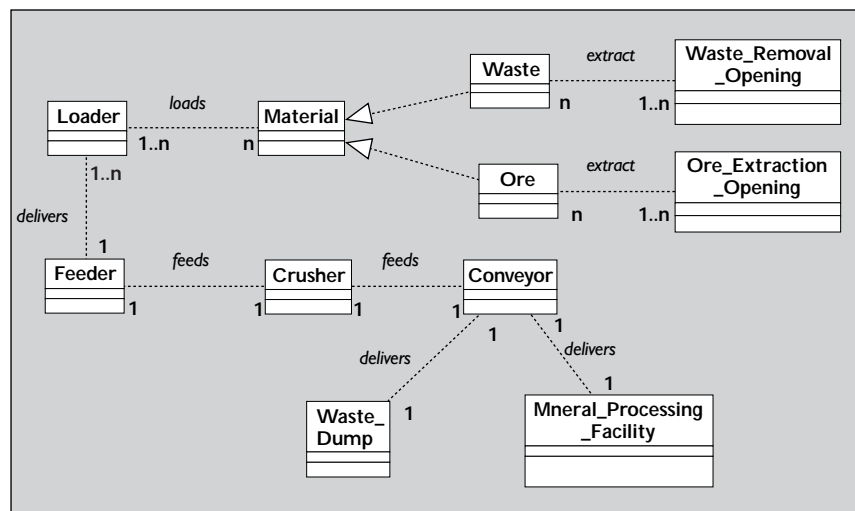


Figure 2b. Class diagram of conventional model for pipeline transportation.

Thinking Objectively

are the plan to reach those goals and IOs are the objects that perform the work to realize the plan [1, 2]. The sequences and logical relationships among those objects are clear. By tracing the objects top-down, we can test and verify the use of every class in the diagram systematically. We will address the issues of measurability and testability in future columns.

Although the stability approach requires a high level of discipline in analysis and a high level of rigor, this is inevitable in any approach that tries to reduce maintenance costs. The three-layer method for identifying EBTs, BOs, and IOs is central to the Stability Approach. System designers can provide a precise, well-structured, adaptable, and maintainable solution for a problem only if they understand why the objects are necessary to the problem.

Conclusion

The common ideas of conventional models try to describe the problems and solutions using real-world objects and by providing a mechanism to guide the subsequent programming process. However, in a stability model, the EBTs and BOs answer the questions: “Why do we build this system?” and “How does the system achieve its goal?” Thus, the key to stability modeling is to identify aspects of the environment in which the software will operate without change, and to cater the software to these areas [1].

In conventional models, engineers often conclude the analysis

when a solution to the problem is identified. On the other hand, the stability model more accurately reflects a designer’s reasoning and the underlying process used in the analysis and exposed through EBTs and BOs. Therefore, we conclude that stability models are easier to understand and evaluate.

While stability models demand a greater investment in analysis, our practical experience has shown that, when used wisely, the savings in development and maintenance costs can more than make up for the additional time spent during analysis. We also believe the stability approach has potential to reduce or eliminate the cost of the reengineering cycles commonplace in software engineering projects. We view the software stability approach as an essential refinement to existing OO analysis and design processes. Moreover, we do not note any added complexity in SSMs. Instead, our study suggests SSMs are concise, elegant, and inherently adaptable and extensible. **□**

MOHAMED FAYAD (fayad@sjsu.edu) is a professor of computer engineering in the Department of Engineering at San José State University, San José, CA.

SHASHA WU (shwu@cse.unl.edu) is a Ph.D. student in the Department of Computer Science and Engineering at the University of Nebraska, Lincoln.

REFERENCES

1. Fayad, M. Accomplishing software stability. *Commun. ACM* 45, 1 (Jan. 2002), 95–98.
2. Fayad, M. and Altman, A. An introduction to software stability. *Commun. ACM* 44, 9 (Sept. 2001), 95–98.