# Article Critique: “Anything You Can Do, I Can Do Meta.”

Jose Sandoval, B.Math

Wilfrid Laurier University

BU655 Management of Innovation and Technology Transfer

Dr. Hamid Noori

March 6, 2007

According to Scott Rosenberg (2007) in his article "Anything You Can Do, I Can Do Meta" Charles Simonyi is probably the most successful software programmer of the Information Technology industry, if his accomplishments are measured in monetary reward (Miller & Serafin, 2006) and the number of people using his applications (Microsoft's Word and Excel).

Simonyi is in the process of introducing a new and radically different way of building software called Intentional Programming (through his company Intentional Software), which is based on his research work published in 1995, while a Microsoft Corporation employee, "The Death of Computer Languages, the Birth of Intentional Programming" (Simonyi, 1995). Although there is no commercial product release date scheduled, early reviewers of the methodology are optimistic about the potential positive impact it could have in the professional software development world (Rosenberg, 2007).

Intentional programming treats context, problem definition, and application development as one. In other words, engineers and business experts analyze a given problem, create business logic flows, and then pass the created set of rules through a *generator* to create a fully functional, defect-free application, with minimal programming required (see Exhibit 1).

In general, software programs are composed of lines of code that are translated into streams of 0s and 1s, which contain sets of formalized rules based on end user requirements, so that digital computers can execute them. Over the years, layers of abstraction have been added to the software development process, which have given rise to different generations of programming languages (Schach, 2002). With each generation the goal has been to separate the programmers from binary bits to facilitate the creation and maintainability of software by using human readable codes. But with all these advancements in programming languages and software methodologies, the promise of delivering defect free application is still looming and billions of dollars, and even human lives, are lost because of defective software (Rosenberg, 2007).

Simonyi believes that adding one more layer of abstraction, in the form of a *meta* framework (intentional programming), will allow engineers and domain experts to better create defect free, large scale software systems. In addition, the lack of actual lines of code will let non-programmers to maintain and enhance existing applications created with this new meta framework. More importantly, the intention or knowledge gained while building a system will be part of the end product, in contrast to existing software engineering practices where all domain knowledge gained in the typical requirement, design, and implementation phases is only preserved in the form of functional and design documents (Rosenberg, 2007).

Even though Simonyi has the resources and the clout to bring intentional programming into the market, there are obstacles to overcome before becoming the norm—if it does at all.

First of all, one of the main draw backs of his solution is that the introduction of intentional programming into any enterprise, whether large or small, will require a drastic change of operations and a large investment in new tools and staff training. The change will not only affect software solutions providers, but customers will also need to change the way their software needs are defined and implemented. For example, the creation of a banking system will not only require the solution providers' engineers to create contextual application, but financial institution's staff will have to become domain experts to serve as application builders. Changing operations in both segments of the industry will prove very costly, indeed.

Second of all, the use of Simonyi's new methodology requires that business rules be codified in some way, thus leading to the introduction of a new higher level, intentional programming specific language in the form of a do-it-all meta framework. This solution does hide the complexity one layer above the action of coding, however, Rosenberg correctly points out that "grand schemes [of this nature] have seldom worked. Every previous innovation introduced as a complete solution to software's woes has ended up providing no more than

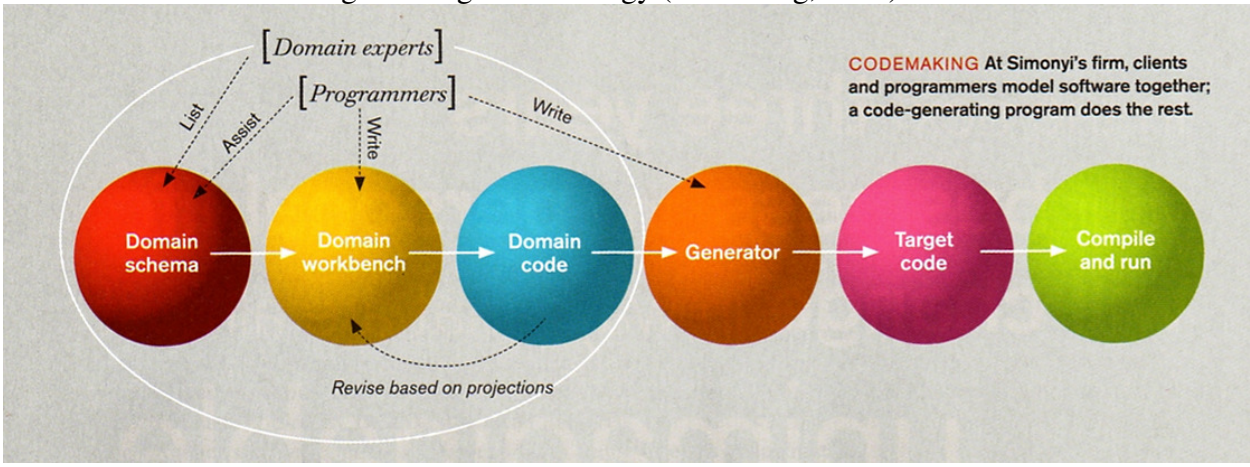
modest, incremental improvements" (Rosenberg, 2007). Moreover, adding a new framework to the engineering process introduces a new set of complexities, thus masking the problem further, unless the intentional programming generator creates a perfect application from the beginning—and this is not a very realistic expectation.

Finally, intentional programming introduces the issue of proprietary technology into the mix of information systems engineering. The customers (of solution providers) will need internal domain experts to be well trained in the methodology to be able to engineer one-of-kind applications, as explained above. This will create large switching costs when a new and better methodology becomes available—and new methods do come along. Moreover, large information systems are not typically built with one proprietary technology. On the contrary, large applications require ongoing relationship between software, hardware, and support vendors throughout the application's life cycle.

Simonyi's intentional programming seems to be headed in the right direction; however, for his disruptive technology to be well accepted throughout the industry, the benefits will have to outweigh the large initial investment. It can also be argued that the software industry is not mature at all (Evans, Hagiu, & Schmalensee, 2006) and may not be ready for such a radical change. As John Naisbitt said, as quoted by Blake L. White (1988), "Change occurs when there is a confluence of both changing values and economic necessity, not before." In this context, even though there are issues while building large scale software applications, the fact is that existing methodologies are evolving with the times. Most importantly, software application, whether buggy or not, seem to be "good enough" to make our society function. Therefore, not until the field of software engineering is in a major crisis, new discontinuous innovations (Moore, 1995) such as intentional programming will likely become the dominant design of the industry.

## Exhibits

Exhibit 1 – Intentional Programming Methodology (Rosenberg, 2007)




## References

Evans, D. S., Hagiu, A., & Schmalensee, R. (2006). *Invisible Engines: How Software Platforms Drive Innovation and Transform Industries* (pp. 81-113). Massachusetts: The MIT Press.

Miller, M., & Serafin, T. (Eds.) (2006, September 21). *Special Report: The 400 Richest Americans*. Forbes. Retrieved February 26, 2007, from http://www.forbes.com/lists/2006/54/biz_06rich400_Charles-Simonyi_VMOW.html

Moore, G. A. (1995). *Inside the Tornado*. New York: HaperCollins Publishers, Inc.

Rosenberg, S. (2007). Anything You Can Do, I Can Do Meta. *Technology Review, 110(1)*, 36-48.

Schach, S. R. (2002). *Object-Oriented and Classical Software Engineering* (5th ed., pp. 437-440). NY: McGraw Hill Companies, Inc.

Simonyi, C. (1995). *The Death of Computer Languages, the Birth of Intentional Programming*.

White, B. L. (1988). *The Technology Assessment Process: a Strategic Framework for Managing Technological Innovation* (p. 36). Connecticut: Greenwood Press, Inc.