ISSUE 35 | APRIL 2010

SOCIETY OF ACTUARIES Technology Section

CompAct

ELECTRONIC NEWSLETTER

CONTENTS

QUICK LINKS

Share Print Article Search Back issues

HORSES FOR COURSES

by Phil Gold

Editor's note: This article originally appeared in the January 2007 issue of CompAct.

We live in an environment where the word open has positive connotations, while closed has a negative feeling to it. Linux is open source, and Windows is proprietary or closed. So Linux must be better, right? Well, for some people it is, and if you read the Internet blogs, there's no contest. Yet Windows has the bigger market share. How many of us are running on Linux today? Strangely, Apple's OS/X gets even better reviews and that is a closed system. So let's keep an open mind.

There is no universal answer that open code is better than closed code, or vice versa, although it seems almost an item of religion for some people. You have to look at the requirements of the application, the quality of the vendor, the size of the organization, initial and ongoing costs of each approach, corporate governance control requirements, the rate of change in the environment and the availability of skilled resources.

I am a software developer and I have chosen an approach closer to the closed end of the spectrum than the open end. I thought you might like to know why I made that decision, and what I have done in my application to meet the requirements for flexibility that many claim can only be satisfied by open code.

The first problem is to define my target market. Let's say my target

Section Specialist

Sam Phillips, Staff Editor

market is all life actuaries, everywhere. I am writing a system to perform all manner of actuarial calculations, first to support the requirements within my own country, then internationally. By far the easiest solution for me is to write an open code system. All I need to do is develop a nice framework, probably based on Excel or something that looks like it, add some database support and a report writer, then prototype some typical products and let the client or a consultant worry about adjusting the sample code to fit the real world products. Then I could look forward to a lucrative stream of consulting assignments to implement and maintain your systems. In fact, I must be nuts not to have followed this model. Why? Because the alternative is to code every possible combination of product features and regulatory requirements myself, as well as user-specific methodologies and approaches, and that would take forever. And yet, masochist that I am, that is indeed the path I chose. I'll tell you how later. Right now, I'll concentrate on why.

Why take the closed code route?

- Because I've been down the open code route before, in three companies. In every case each new product resulted in a new model with new source code, incompatible with other models in the company, and a variety of errors because of the lack of a proper testing environment. I want a reliable universal model, not a collection of independent models.
- Because I believe actuarial resources are scarce and expensive, and they should be employed on real actuarial problems, not developing software. Those actuaries that want to develop software should come and work with me–I'll need them for sure.
- Because despite the unique characteristics of each company and product, there is a lot more common ground between us than elements that separate us.
- Because I like a challenge. When someone tells me it can't be done, that's when I get interested.
- Because I was working in a reinsurance company, where we required a quick turnaround on each new product and the volume of such products prohibited the down time of developing a new model each time around.

Why persevere?

- Because the senior management of my company encouraged my efforts.

- Because my initial efforts were met with surprising success in the market.
- Because I found partners and employees who shared my philosophy.
- Because our clients gave us encouraging feedback.

Now the golden rule in software is do not bite off more than you can chew. The whole industry is tarred by those who promise the world and don't deliver. We developed a different philosophy. We never promised something unless we were certain to achieve it, and we encouraged potential customers to simply try what we had right now, and see if it would be useful to them. I would urge all in the industry to follow this model. This way, you get clients who trust you, and recommend you to their friends. We have lost new business along the way by refusing to make aggressive promises, but I am convinced it is the right way to proceed.

We proceeded by concentrating on particular market segments. We could not be all things to all people, especially at first. So we built out our portfolio gradually–first conventional products, then UL, then Par, then Disability, then Assets and so on, at the rate of about one new product line or module per year. We started off with just one country, then two, until today we operate on five continents.

Most of all, we concentrated on keeping our current customers happy. When you write closed code, you are taking on the responsibility of providing good service and fast response time. In our system there is only one code base and everyone gets the same software. So every client gets the benefit of each new feature no matter where the request came from.

OK, there's a problem right there–what about secret new features you don't want your competition to know about? This does happen, although as you know, there are few secrets in this industry. Suffice it to say there are various ways to solve this problem, and we can build in enough flexibility so that proprietary product features will not be given away by the software.

This is where the closed code approach pays off in a big way. If you have just one code base, then you can have many users reviewing and validating the calculations. It helps enormously when regulators and consultants review your software and during acceptance testing at each new client. If you have something wrong or missing in your code, you're going to find out and have a chance to fix it. This simply is not the case for open code. Think for a minute about the SOX

implications here, their importance simply cannot be overestimated. Actually this is not a consequence of closed code but of common code. By going open code you simply preclude this level of scrutiny.

If you ask actuaries what is the biggest problem they face with their software, the most common answer will probably be the problem of keeping it up to date, the problem of conversions. Let's examine how these work on a completely closed system and a completely open system. On a completely open system, the vendor can really only provide changes to the framework and some new sample code. Changes to the framework must be limited or they will disrupt the current implementations. So the onus is on the developer to come up with the perfect framework for all time on day one. Let's be honest here. How many system architectures from 15 years ago are currently state of the art? If you got something wrong on day one, you can't always fix it later because users will have built their application around your architecture–change it and it breaks their applications. Now in the closed code context, we can provide automatic conversions of user models no matter what changes we make to the architecture. We have changed from DOS to Windows, from Basic to C++, from FoxPro to JET, and from a separate system for each country to a unified system, all without breaking the users' applications. Sure it takes a lot of work on our part, but the advantages are overwhelming. Users get a system that can be kept up-to-date with all the latest technology and functionality without massive conversion problems. They can upgrade in hours and not months. This is one of the biggest justifications for the closed code approach.

Now I promised to tell you how we tackle the need for flexibility, given we don't have infinite resources. Actually, it really does take an enormous effort to build in all the features clients require, and we have a very large and highly skilled workforce here we would not need if we were an open code shop. We have found that in most areas of the actuarial system we can provide enormous flexibility through the switches, scalars, tables and objects we have built up over the years, some at our own instigation and some to meet user request (about a 50:50 split). But there are some types of logic that are very hard to accommodate in this way. Take policyholder behavior for example, or crediting rate strategies or experience refunds.

Some closed code systems are really closed down tight. Others allow you insertion points, and with the aid of a compiler or by using a non-compiled language, they allow you to change the source code. The first option seems too rigid to me, although it is what we offered

for a number of years because we felt the second option simply negates most of the advantages of closed code systems. Then one day our developers came up with a third way that provides the advantages of open code wherever you need it most, but preserves the integrity of the source code and allows for full automatic conversions between software releases. I won't go into too much detail, but the breakthrough involves treating user code as data input to the system, and some very sophisticated use of the .NET framework. The cost of having this type of expertise in house may be prohibitive for the end user.

If I were starting over today, would I do the same again? I'd have to say that from a financial and marketing point of view maybe not. It is cheaper to develop an open code system and probably easier to sell. But when I think of the client's best interest, I would have to say yes. For less ambitious software projects where the scope is more focused, the balance may be completely different. Each case should be taken on its own merits. So the golden rule is there is no golden rule. Choose instead the right horse for your particular course.

Phil Gold, FSA, MAAA, FIA, is a founding partner of GGY AXIS. He can be contacted at phil.gold@ggy.com.

Actuaries
Risk is Opportunity.®

475 North Martingale Road, Suite 600 Schaumburg, Illinois 60173
Phone: 847.706.3500 Fax: 847.706.3599 www.soa.org