

Making Software Hard — to reduce risks!

ShouldersCorp
www.shoulderscorp.com

Architects develop drawings and physical models to help their customers get excited about a new construction project. Software architects and project managers have the mistaken impression that architectural models are developed for the benefit of the construction crew. Software projects need physical models for a different reason altogether, risk management. Keep reading, we'll explain.

ShouldersCorp has developed a technique using physical models of a developing software system, which achieves the following benefits:

- Display the current status of a developing software system
- Manage "creeping featurism" during the course of a project
- Aid new team members to understand the developing software
- Improve the quality of the software design
- Evaluate the rate of development progress relative to the published schedule
- Revise completion date based upon rate of development.



The physical models must be combined with metrics, estimation tools and objective evaluation techniques. With this combination, the company has an unbroken record of 9 projects on time, on spec and on budget. The company has figured out how to package the essence of its management technique and offer it to any project. The cost is low; the benefits great.

The ShouldersCorp management tools and support personnel will cost about 2% of a development project. The power is its simplicity. Hiding behind the simplicity is 30 years of experience building, managing and rescuing complex software projects -- some involved over a thousand programmers, others only one!

How does this technique work? Let's take an example using a modern object-oriented programming language such as Java. Variations of the technique have been developed for procedural languages such as COBOL, markup languages such as HTML, and report writers such as Crystal Reports. The process begins with the review of a project plan to determine key metrics that have been shown to be objective predictors of software development effort and schedule.



For a Java project these include the number of:

- Core business classes
- Utility classes
- Database tables
- Elements in the data dictionary
- Business rules
- Screens
- Roles (types of users)
- Interfaces to legacy systems



With this information ShouldersCorp uses its proprietary estimation tools to estimate the effort and schedule for the project. This estimate may or may not agree with the official project estimate; quickly they will converge.

Once the names of the classes remain stable for a few days, its time to build a physical model of the system. Since Java is an object-oriented language, which supports inheritance, we have found that an inheritance tree is the ideal physical representation of the developing Java system. We have also found that a class is the ideal unit of work -- it's about 20 procedures or subroutines, if you don't happen to be familiar with Java. The sheer size of the physical model is itself revealing. The largest project we have done involved 1758 Java classes. 100-day projects often result in less than 100 new Java classes.

Once we know the number, the structure and the names of each class, we use Styrofoam balls, wooden dowels, a glue gun and a label maker to build a physical model of the emerging software system. The resulting model is hung from the ceiling in the development area so that everyone can see what they are building. But we do so much more. The Styrofoam balls are proportionate in size to the number of methods in each class. Each class or ball is labeled with the name of the class. Then we use ribbons to designate the current status of each class. Ribbons are tied around every class to designate its current status.

- Red** **designed**
- Blue** **code complete**
- Green** **documented**
- Gold** **all test cases complete**

The class structure and the ribbons are updated every day, so that anyone on the project can see the current status of the project at any time. We have had clients install a web cam so that all stakeholders can see the status of the project from their workstation. But

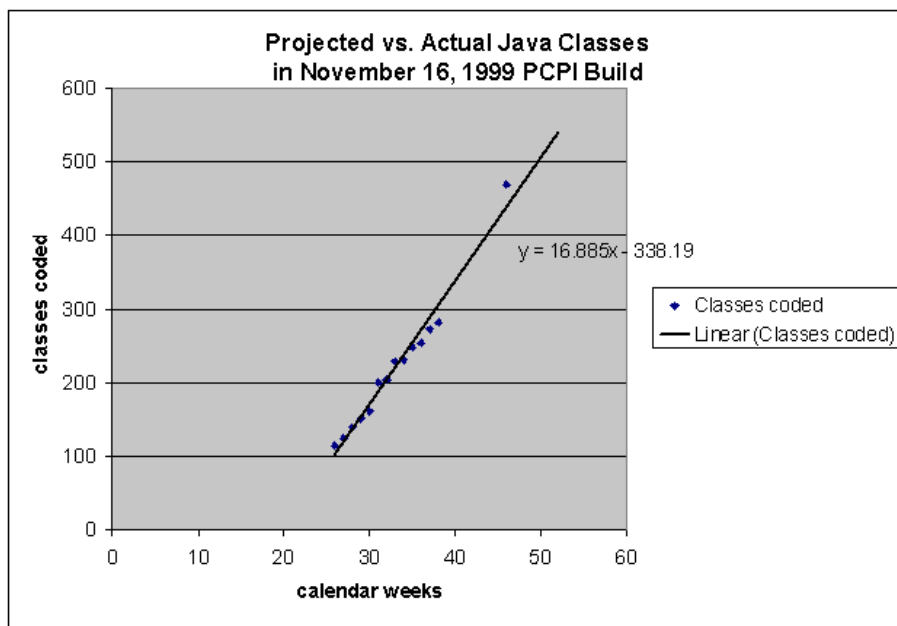


there is real value in encouraging stakeholders to walk into the development area and experience the 3-D model on a regular basis while talking to the development team. It's incredibly difficult to stand in front of a structure containing 75 components, none of which have been tested or documented and imagine that the project will be finished anytime soon!

A project manager can walk up to the model and see the current status of the project. If almost every class has a gold ribbon around it, completion is close at hand. If most classes have only yellow ribbons, there is a lot more work to be done.

The physical model is also augmented with objective metrics, the most important of which show rate of change. If the project has been able to complete 10 classes per week for each of the last 6 weeks, it's a reasonable guess that the project will be able to complete no more than 10 classes per week for the next several weeks. If there are 40 classes to be developed and the schedule shows 6 weeks remaining, the project should be in fine shape. But if there are 40 classes remaining and only two weeks left, a delay should be expected. More intense management and longer hours can have a small effect, but only a small effect. Miracles don't happen in software projects. Disappointments are commonplace.

Rate of change metrics are posted on a wall (or bulletin board) near the physical model and updated at least daily. These rate of change charts are also posted on the project website. Rate of change metrics have proven to be very reliable predictors of future performance, e.g., completion dates. The following chart shows an actual prediction on a major project. Almost three months in advance the prediction was off by only a day! If you have estimation tools more accurate than these, you don't need our services.





With good estimation tools, a physical model, good communication among all stakeholders, and objective measures of progress, even complex software projects can be delivered on time, on spec and on budget. It requires 2% of a project's budget to manage the development risk in this way. Since recent statistics show that complex software projects fail 40% of the time, a 2% investment for more effective project risk management is money well spent.

Making Software Hard...

Every project manager will immediately ask, "why don't you show the physical model using virtual reality or some such new technology?" We have our reasons.

On the first Java project that we built a physical model of the developing system, it so happened that the CFO of this 200-person company had an office nearby. He walked by the project space to get coffee. CFO's like free coffee. It didn't take him long before he asked "why we were building a structure out of Styrofoam balls and wooden dowels, given our billing rates". Then he began to think of each of these "balls" as a person week of effort and he quickly learned that we're not done until all the "balls" have gold ribbons. Now he could get a project update every time he refilled his coffee. He got so involved in the project that he even won the prize for finding the most errors in the software during a release to the company for a final weekend of testing before the public release. He had learned a lot about why software projects are so hard and gained a real appreciation for the effort we had to put in to make the project happen on the appointed day with the agreed functionality.

There is no chance a CFO would make a daily visit to a website to look at the status of a project, unless that project was spending way too much money already. It so much easier to run a project with the CFO on your team than have her challenge every expenditure!

There is another reason for having a physical model. If the risk management team arrives on a Monday morning and discovers that 23 new classes have been added to the system, then a structure containing 23 new "balls" will have to be added. Now all of a sudden, the whole structure will look differently. Everyone walking by on their way to the coffee pot will ask, "what happened this weekend?" Detecting such a major change quickly and taking decisive action is exactly what we want to happen.

A final reason for having a physical model of developing software is that it's a great tool for training new team members. Several people can easily stand around a physical model of the developing software. An expert can explain the major components, how they fit together and how the new person's software fits into the overall system. This is crucial in order to maintain the integrity of the system architecture. It is easily done with a physical model; it's awkward to do with a computer-generated model.



We have no problem with "publishing" the physical model of the software so that it can be viewed remotely. But it augments the physical model; it does not replace it. Remember glue is cheap and unexpected software is very expensive.

In summary, we have found that “making the software hard (physical), helps to reduce risks”.

Professional Risk Management

What ShouldersCorp offers is professional risk management services for software projects. These services are tailored to the size and complexity of any given project — and backed by a corporate commitment to total quality management.

On complex assignments involving numerous projects and contractors, ShouldersCorp's risk management services must begin during design and may include:

- Schedule, budget and technology risk evaluations
- Constructibility reviews based upon analogous projects
- Accuracy checks on cost and schedule estimates
- Considering project alternatives regarding costs and risks
- Structuring projects for lowest cost or shortest schedule construction
- Monitoring and coordinating daily construction activities

ShouldersCorp's risk management personnel are on-site software professionals with a track record of success.