Jay Almarode

Putting It All Together

SCENE 1:

A well-lit conference room in the offices of ABC Corporation. Facing intense competition and a changing marketplace, the IS department has been given the directive to re-engineer its software systems to be more flexible and easier to maintain.

Pat (development manager): Thanks for gathering on such short notice, team members. I've got good news and bad news. As you know, today we gave a demo of our application prototype that we built in Smalltalk to the head of development. The good news is that management was very impressed and decided to give the green light to build all our new applications in Smalltalk. The bad news is that while giving the demo, the VP of development walked in and thought we had already finished the application.

Chris (engineer): What gives? I hope you set him straight.

Pat: The VP saw finished screens with apparently real data and thought the application was running. However, I explained that this was a prototype implemented in single-user Smalltalk, and that the display was using dummy data stored locally in the image.

Bobby (another engineer): So then what happened? What made management sign off?

Pat: I think they were impressed with how quickly we built the prototype. But I think the real kicker was when the VP looked at the order-entry screen and said that we were missing a field showing compatible part numbers. I opened a browser and added an instance variable to class OrderEntry, then added a widget to display it. In less than two minutes, I had the new screen. The VP's mouth just dropped.

Terry (another engineer): Did you save the image so you can check that code into the repository later?

Pat: Of course. So now that we've got the go-ahead, the real work begins. I'm afraid the expectation level is high on this, so we've got to get organized right away. Right now, our office is organized along functional areas. Each functional area has its own application-development team. I've been reading the latest project management and methodology books, so I've assigned developers from each functional area to design and build our common business object model.

Bobby: You've sent everybody to training classes, but do you think we're ready for prime time?

Pat: I have the highest confidence in all of you. However, I've also contracted mentors from Objects-R-Us to help us out. They'll be on-site for both design and implementation. They're Smalltalk gurus and should get us through any rough spots.

Chris: Yeah, but this is a large system. We're talking over twenty applications, hundreds of users, and lots of objects. How are we gonna make sure that we scale?

Pat: Good question. Our charter is to build objects that span the enterprise. To make sure we know what we're getting into, I've assigned you to teams to look at the following issues. *(Pat goes to the white board and begins writing.)* One team will look at overall system-performance characteristics. We need to get a handle on object counts and required response times. Another team will look at the system configuration and architecture. This includes hardware and fault-tolerance issues, as well as system administration. The third team will look at application-design issues. They should make recommendations for applications to handle multiuser issues and meet performance requirements.

Terry: These are big responsibilities. How are we going to meet them?

Pat: Obviously, these teams do not work in isolation. I expect a lot of cross-communication between these teams, as well as interaction with the individual application-development teams. The three teams I've described and the common business object team have system-wide visibility.

SCENE 2:

A small cubicle filled with books, hardware, and Dilbert cartoons.

Terry: Hey Bobby, got a minute? I was wondering if you could take a look at this questionnaire I created? I'm try-

Using Smalltalk since 1986, Jay Almarode has built CASE tools, interfaces to relational databases, multi-user classes, and query subsystems. He is currently a senior software engineer at GemStone Systems Inc., and can be reached at almarode@gemstone.com.

ing to capture the overall system characteristics. I figure I'll get answers for every application, then try to accumulate them for the system as a whole.

Bobby: Hmm, looks interesting. What is the purpose of asking for the number and size of objects?

Terry: I'm trying to come up with some estimate of the object repository size. I've got to make sure we've got enough disks for all applications.

Bobby: Well, just make sure you account for growth. Remember when forecasting built their new models last year, and then didn't have enough space to store them? Also, how can someone tell how much garbage they're producing?

Terry: The mentor from Objects-R-Us gave me this neat little goodie. It should be used only during development, but it tells me which objects were created and which would be collected when a block of code is executed.

```
classmethod: System
newObjectsAndGarbageWhile: aBlock

" Return an array of two sub-arrays: the first
containing objects that were created, and the
second containing objects that became eligible
for garbage collection during the execution
of the given block. "

System _generationScavenge.
System _hiddenSetReinit: 31.  "ObjsCreated"
System _hiddenSetReinit: 32.  "ObjsDisposed"
System _enableTraceObjs.

aBlock value.

System _generationScavenge.
System _disableTraceObjs.

^ #[ System _hiddenSetAsArray: 31,
    System _hiddenSetAsArray: 32 ]
```

(Chris enters the cubicle.)

Chris: Hey, have you heard the latest? We've got approval to go three-tier. Our arguments convinced Pat that we needed a real application server. Now we can implement our applications in whatever vendor's single-user Smalltalk we want, and still share Smalltalk objects on the server.

Bobby: How did you convince them?

Chris: Well, they recognized that we had to access legacy data, and already knew that the object-to-relational mapping was nontrivial. We basically told them that we couldn't afford to send all that data over the wire to each client, perform the mapping to create objects, and then map changes back to SQL updates. Plus, we couldn't guarantee security or fault tolerance on the client.

Terry: Great. This should make our business object modelers happy. Now they have a single place to maintain their objects. So how are those people doing?

Chris: I heard they've had to do a lot of work. Our original demo did a good job of separating interface from domain objects, but the domain objects mixed application-specific with general behavior. They've been busy figuring out which objects belong on the server and which belong on the client.

Bobby: So how are they figuring this out?

Chris: Our mentor has given us some tips to help them. It's really common sense anyway. You know, large collections, shared objects, secure objects, recoverable objects; they all live on the server. Speaking of recoverable objects, how's it going with system configuration, Terry?

Terry: We're just getting our hands around it. We've come up with an architecture diagram showing which machines will house shared-page caches for the clients. Our plan is to assign certain applications to certain server machines to try to spread out the load.

We're splitting the repository into three raw disk partitions for performance. We'll also dedicate one disk to transaction logs, and schedule a job to run nightly, compressing and copying the logs to tape. We still haven't figured how often we'll checkpoint the repository, though. We haven't heard back from all application areas on their fault-tolerance needs. We asked each area to spell out how

User access:
What is the total number of users?
What is the average number of users logged in at any given time?
What is the maximum number of users that may be logged in?
How many different geographic locations?

Number and size of objects:
What is the total number of objects?
What is the average size of an object?
What will be the largest objects?
What are the sizes of the largest collections?
How much garbage is produced per transaction?

Transactions:
What is the expected transaction rate ...
per day, per hour, per minute, per second?
What is the peak transaction rate?
What is the duration of the longest transaction?
How many objects are read during each transaction?
How many objects are written during each transaction?
How many objects are created during each transaction?

much downtime is acceptable, and when, but I think they're still working on design issues.

SCENE 3:

In the nautilus gym at ABC Corporation. (What? Your company doesn't have a gym?)

(Terry walks over to Chris, who is puffing away on a lifecycle.)

Terry: Hey Chris, how's it going? Have you solved all of our application design issues?

Chris: We're making progress (puff, puff). I've been talking to the chief designers of all the applications, and we've identified the shared collections. We've decided to use some special multiuser collections to reduce the chance of concurrency conflicts. We still have to worry about other chances of conflict, so we've been looking at when to lock objects and when to design for the possibility of conflict.

Terry: Oh really. How do you make that determination?

Chris: I asked developers for each application to define their transactions, and to identify which shared objects would be written for each kind of transaction. We categorized the transactions by their priority, the probability of conflict, and the impact of transaction failure. From there, we began planning our strategies.

Terry: Hmm. . . interesting. How did it go?

Chris: As you can guess, some designers knew exactly which objects they were modifying, while others didn't have a clue. Fortunately, we had the capability to view the set of written objects at transaction boundaries. We ran some test cases and saw which objects were being touched. This really opened up some developers' eyes and we improved the code as a result.

Terry: I'd like to get some of your instrumentation code. So what strategies do you come up with when you don't lock objects?

Chris: We've created a framework to keep a log of important object modifications that should be replayed in the event of concurrency conflicts. Basically, these are temporary objects that are created during the life of a single transaction. We found this was best wired into the application rather than into the business objects, because it is the application that defines a transaction.

Terry: So how do you make sure these objects remain temporary?

Chris: We found in the manual something called 'transient session state', from which you can reference objects so that they do not get garbage-collected, yet it doesn't cause them to be part of a committed state. Of course, if you reference these objects from some other committed object, they will become persistent.

Terry: I don't understand. How does this all fit together?

Chris: For certain operations, we create objects that encapsulate the modification to a business object. We hang these objects off of a transient session state. If the transaction should experience conflict, we can abort the transaction, then replay the modifications. We had to design this carefully, because sometimes concurrency conflicts are a good thing. Our framework allows us to perform validation after the transaction is aborted, to make sure that conditions still hold to replay the modifications.

Terry: Very clever. I hope you're making this available to all applications.

SCENE 4:

Six months later, in the well-lit conference room.

Pat: Thanks for gathering on such short notice, team members. I've got good news and bad news. As you know, we just deployed our last application and the user response has been positive. There were a few glitches along the way, but we managed to hang in there and deliver all that we said we would. We learned a lot along the way, such as making sure hired consultants really know Smalltalk, and planning for schema modification after applications have been deployed. The bad news is that management has been sufficiently impressed with our deliveries that it wants us to do the same for the international offices, but in less time. In addition, we'll need to replicate objects across distributed servers for local availability. I'm afraid the expectation level is high on this, so we've got to get organized right away. So here is what we're gonna do ... S