| EAA-dev Home |

WORK-IN-PROGRESS: - this material is still under development

Temporal Patterns

Summarizes various patterns that you can use to answer questions about the state of an information in the past. These include questions of the form "what was Martin's address on 1 Jul 1999" and "what did we think Martin's address was on 1 Jul 1999 when we sent him a bill on 12 Aug 1999".

Last significant update: 16 Feb 05

Things change. If we store information about the world this may not be a problem. After all when something changes one of the great values of a computerized record system is that it allows us to easily update a record without resorting to liquid paper or retyping pages of information.

Things get interesting, however, when we need to record the history of the changes. Not just do we want to know the state of the world, we want to know the state of the world six months ago. Even worse we may want to know what two months ago we thought the state of the world six months ago was. These questions lead us into a fascinating ground of temporal patterns, which are all to do with organizing objects that allow us to find answers to these questions easily, without completely tangling up our domain model. Of all the challenges of object modeling, this is both one of the most common and most complicated.

The simplest way to solve this problem is to use an Audit Log. Here you are concerned with keeping a record of changes, but you don't expect to go back and use it very often. Therefore you want it to be easy to create and be minimally intrusive upon your work. When someone needs to look at it, then you expect they will have to do a lot of work to dig out the information. If they don't need to do it very often, and don't need the resulting information quickly, then this is fine. Indeed if you're using a database, it's free.

When the information has to be more accessible, then you need to work a little more at it. The most common pattern that people use in this situation is Effectivity. This simply marks an object with the time period that it is considered valid (Figure 1). Then as you manipulate your objects you use this time period to get the right object at the right time.

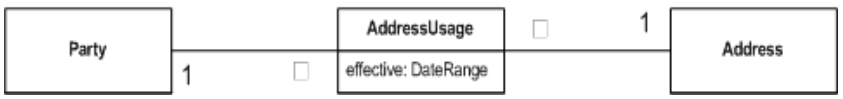

Figure 1: Using an explicit date range to show effectivity.

The problem with Effectivity is that it is obvious and explicit. Everyone using the object in all contexts has to be aware of the temporal aspect. This can complicate a model considerably. So if temporal issues are pervasive then it often makes sense to hide them when you don't need them, and make them more convenient to work with when you do.

If you have just a few properties of objects that need easily accessible temporal information, then you can use Temporal Property on those objects. The essential point of Temporal Property is that you remove the obvious effectivity date, instead you use what looks like a regular property, but with an accessor that takes a date as an argument. This allows you to ask "what is the value of this property at April 1st 2000".

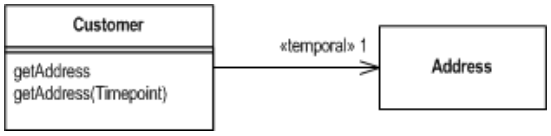

Figure 2: Using a temporal property for the address..

The essence of Temporal Property is this notion of having an accessor that is parameterized by date that frees the users of this property from having to navigate through a bunch of objects with Effectivity. Therefore Temporal Property and Effectivity are not mutually exclusive patterns. You might use address usage objects (using Effectivity) to implement a Temporal Property interface. If the address usage objects carry other responsibilities you may provide methods that access the address usages for those clients that need those other responsibilities and a Temporal Property interface for those who find that more convenient.

Temporal Property introduces the notion of referring to things with a time-based index, but takes it only so far as a single property. If you have many temporal properties on an object, things can become clumsy. There are two ways of dealing with this. The first is to use Snapshot, which gives you an object that refers to the state of the real object as at a point in time. Thus if you want to ask a lot of questions of a customer as at April 1st 2000 you ask for a Snapshot of that customer for April 1st 2000 and then ask for values of properties without having to keep repeating the date.

Figure 3: A snapshot shows a view of an object as at some date.

Snapshot allows you to have a view of an object at a date, but sometimes you really want to think of an object going through explicit versions as changes are made through time. This desire leads to Temporal Object. Temporal Object comes in many forms, but the most straightforward form has two classes as in Figure 4. The contract class is the one that is usually referred to by other objects as it represents the contract through time. It contains a collection of versions which record the state of the contract at each change. This allows people to refer to either the unchanging notion of a contact, or a specific version at a particular point in time.

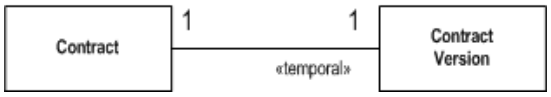

Figure 4: Temporal Objects have an explicit history of versions, so that each change leads to a new version.

In practice I see Effectivity used widely, but often that is because people aren't familiar enough with Temporal Property and the other more sophisticated patterns. Patterns like Temporal Property and Temporal Object work well because they hide much of the mechanics of temporal behavior. However Effectivity still is important: it is the right choice when people want an explicit object that is only valid for a particular time period.

Dimensions of Time

Just as I've described above, time is a pretty challenging notion in modeling. However I've skipped over the most awkward aspect of temporal models. We've all learned, if only from bad Science Fiction books, that time is the fourth dimension. The trouble is that this is wrong.

I find the best way to describe this problem is with an example. Imagine we have a payroll system that knows that en employee has a rate of $100/day starting on January 1. On February 25 we run the payroll with this rate. On March 15 we learn that, effective on February 15, the employee's rate changed to $211/day. What should we answer when we are asked what the rate was for February 25?

In one sense we should answer $211, since now we know that that was the rate. But often we cannot ignore that on Feb 25 we thought the rate was $100, after all that's when we ran payroll. We printed a check, sent to him, and he cashed it. These all occurred based on the amount that his rate was. If the tax authorities asked us for his rate on Feb 25, this becomes important.

In fact we can think that there are indeed two histories of Dinsdale's pay rate that are important to us. The history that we know now, and the history we knew on February 25. Indeed in general we can say that not just is there a history of dinsdale's pay rate for every day in the past, but there is a history of histories of Dinsdale's pay rates. Time isn't the fourth dimension, it's the fourth and fifth dimensions!

I think of the first dimension as **actual time**: the time something happened. The second dimension is **record time**, the time we knew about it. Whenever something happens, there are always these two times that come with it. Dinsdale's pay raise had a actual date of Feb 15 and a record date of March 15. Similarly when we ask what Dinsdale's pay rate was, we really need to provide two dates: a record date and an actual date.

record date	actual date	Dinsdale's rate
Jan 1	Jan 1	$100/day
Feb 25	Feb 25	$100/day
Mar 14	Feb 25	$100/day
Mar 15	Jan 1	$100/day
Mar 15	Feb 25	$211/day

We can think of the two dimensions like this. The actual history is looking back in actual time. If I look at my current actual history, then I see that Dinsdale's pay was $100 up to Feb 15, at which point it increased to $211. However that is the actual history for today in record time. If I look at the actual history for Feb 25, then Dinsdale was paid at $100 from Jan 1 onwards, and the $211 never came into it. Every day (strictly every Time Point) in record time has an actual history. These histories are different as we find out that things we used to think were true are no longer true.

From another angle we can say that each day in actual history has a record history. The record history tells us how our knowledge of that day changed over time. So Feb 25 in actual time has a record history that says that up to March 15, Dinsdale's pay was $100 at which point it reaches $211.

Let's take this example a step further Assume we make the corresponding adjustments in a payroll run on March 26. On April 4 we are told that the employee's our previous information was wrong and that the rate was actually changed to $255 on February 15. Now how do we answer the question "what was the employee's rate on February 25?".

I've seen grown developers bite their own heads off when faced with this kind of stuff. But once you realize that everything comes down to this notion of two dimensions, then things start getting a lot simpler. One way of visualizing this to extend the earlier table

record date	actual date	employee's rate
Jan 1	Jan 1	$100/day
Feb 25	Feb 25	$100/day
Mar 14	Feb 25	$100/day
Mar 15	Jan 1	$100/day
Mar 15	Feb 25	$211/day
Mar 26	Feb 25	$211/day
Apr 4	Jan 1	$100/day
Apr 4	Feb 25	$255/day

If we look at our current actual history (that is the actual history whose record date is today) then we would say that Dinsdale's pay was $100 from Jan 1 and went up to $255 on Feb 15. For the current actual history the $211 rate doesn't occur at all because it was never true. If we look at the actual history for March 26 we would see Dinsdale's pay at $100 up to Feb 15 where it went up to $211. In the actual history for March 26 the $255 rate never happened because we didn't know of it yet.

We can also think about the record history for Feb 25. Now this record history says that the

rate was $100 (on that day) until March 15 when it changed to $211. Then on April 4 it changed again to $255.

Once you realize about the two dimensions then it becomes much easier to think about the problem, but scary to think that you have to implement this kind of stuff. Fortunately there are a number things you can do that simplifies matters when it comes to implementation.

The first simplification is that it isn't difficult to use Audit Log to cope with these changes. All you need to do is record both the record date and the actual date in the log with each entry. This simple exercise is enough to keep any log effective over both dimensions, and I belive is worth doing even if you are only bothered about one of them.

The second simplification is that you often don't want your model to handle both dimensions. The important thing here is to know which you have in your model and which one you are leaving to Audit Log.

If we want to keep a history of things where we want to know how things changed over time, but didn't care about when we learned of changes, we would say that was *actual-temporal*. So if I keep a record of an employee's address I might choose to keep that as a actual-temporal property. For information systems that are there to help on-line query this works well because when you are accessing a database you usually want to know about actual-history.

Record-temporal facts appear when you have a system that does things like producing bills based on the state of objects. These things lead to questions about how the bill was calculated, which leads you to need to know what thought the state of and object was when the bill was calculated. Record-temporal facts can often be compared to a version control system in software where you can go back and say 'what was this file like on April 1?'

Of course there are times when you need both dimensions at once - these are called bi-temporal facts. Essentially bi-temporal information always needs both dates.

Bi-temporality is the full solution, but it's always worth thinking of ways around it. An example comes from bill calculation. If you want to find out why a bill came out to what it was one possibility is to have a fully bi-temporal database. However it's often better to store a detailed trace of the calculations when you calculate the bill. That satisfies the requirement in a much simpler way than a bi-temporal object model.

Updating a Temporal Record

So far I've only talked about temporal information in terms of accessing information, not in terms of updating it. How you allow your updates leads to more decisions, but many of these involve useful simplifications.

In general changing the time record is quite messy. If we have an employee whose rate was $211 from Feb 15 to Apr 15, a fully general update will allow to change any combination of start date, end date, and value. Providing an interface for this is awkward since it requires the client to know a lot about how temporal information works.

The first simplification is that you may have only *additive* changes. An additive change always goes onto the end of the record. An example of an additive change is "make the employees rate $211 effective Feb 15". Although at first blush this looks like you are only removing one piece of data from the mix, the consequences actually work out so that it greatly simplifies the update. Most of the updates that happen in practice are additive, so that can greatly simplify clients. Furthermore you can make any change with some combination of additive updates. While this would get horribly messy on an object with a complicated history, this property can simplify objects with a simple history by allowing you to only support an additive interface.

The second simplification is allowing only current updates. A current update allows changes in the record only with an effective date of today. In general even additive changes can occur at any date in the past or future. A current change requires no date information at all, allowing you to have an interface for updates that is completely non-temporal.

Current updates seem too good to be true, why have temporal information if you only update

it with current updates? The good news here is that record-temporal information can only be updated with current updates. Not just would a retroactive change to record-temporal information break the integrity of the record, there's no situation that requires a retroactive change that can't be handled by the actual-time dimension (unless fraud is one of your requirements). This is a big step as it means that one whole dimension has invisible updates - you only have to worry about record time when querying, not when updating.

Other Readings

Since this is a complicated area, it has generated some other writings as people have explored these problems. The most comprehensive treatment of this area is Snodgrass. His work is based on relational databases and most of the book is based on how to handle these issues in SQL. The problem, however, is the same. Furthermore much of the comments he makes and terminology he uses is the same as the forthcoming SQL standard's material on temporal databases. The book is now out of print, but you can get a pdf from Richard Snodgrass's publications page.

The matter of terminology is one that I'm still not completely happy with. He uses the same two dimensions that I do, but his terms are different. What I call actual time he calls *valid time*, what I call record time he calls *transaction time*. In earlier versions of these patterns I followed his terms, but many people said they found these terms very difficult to follow. As a result I decided to use different terms. Another difference is that he refers to a table that changes over time as *sequenced*. Of course, objects use sequences all the time, not just for temporal purposes, so I've called things that change over time temporal.

[Anderson] is one of the best collections of temporal patterns. Again I've plundered ideas, but changed some terminology. I've used the term *version* rather than *edition* to mean a value for a property or object during some time period. His *Change Log* and *History on Association* patterns are the same as Temporal Property. I see the difference between the two Anderson patterns as more a matter of implementation. His *History on Self* pattern is the same as Temporal Object.

Also in the same PLoPD 4book, you'll find a paper [Carlson et al]written by Andy Carlson in collaboration with myself and Sharon Estepp. This gives earlier descriptions of Temporal Property and Snapshot. It also mentions a pattern *Temporal Association* which I now look at as a use of Effectivity.

Also submitted to PLoP, but not available publicly at the moment is [Arnoldi et al]. This introduces a number of interesting ideas that I haven't fully explored in the patterns here (at least not yet).

Both [Anderson] and [Arnoldi et al] consider using objects other than Time Point to act as indexes into the temporal record. [Anderson] uses a (single-temporal) event, and [Arnoldi et al] use a *perspective:* essentially two timepoints combined into a single object. Although there are things to be said for both of these approaches, I haven't used them here as I feel that using time points does most of what you need, and is simpler to explain.

Significant Revisions

16 Feb 05: Moved these pattern over to the EAA development section.

15 Jan 01: Changed bidimensional terms from valid/transaction to actual/record.

28 Aug 00: First draft

DSL
Delivery
About
Me

Continuous Integration

UML Distilled
Analysis Patterns
Planning Extreme Programming
Signature Series

DSL Catalog
EAA Catalog
EAA Dev
Photos

ThoughtWorks

Blogs
Careers
Mingle
Twist
Go