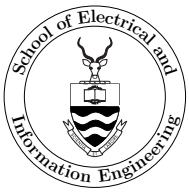

UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG
School of Electrical and Information Engineering
ELEN7045 – SD Methodologies, Analysis and Design

Domain-Driven Design

Data is the Application, or Not?

> I'll say it again, for me, the data is the application. Processes, classes, and code exist in the end only to manipulate the data. Far too often folks get caught up in abstract domain models and diagrams and end up creating nightmares for their fellow developers.
>
> — *Mike Griffin*

> I couldn't disagree more... Well, maybe but not much more. Using the modern SOA paradigm, reuse is aimed not at data but at business logic. The underlying data, in whatever form it may take, is irrelevant. When you build an application that accesses an external web service, it's to add that service's functionality to your application. It's the Borg Manifesto, the service's uniqueness will be made part of a collective (application). Resistance is futile. :-)
>
> Too many developers concentrate on using services as a means to access data and are missing the bigger picture. What good would it do to access Google's data without running their search algorithm on it? The data you receive as results in no way represents the data their algorithm uses.
>
> Data only exists to store the state of some business process. For example, the concept of a customer address only has relevance when applied to some interaction with that customer via that address. If nothing your business ever does involves their physical address, would you still store it? No. You only store the data that your processes need, therefore the data is dependent on the process, not the other way around.
>
> Modeling those processes, no matter which method you choose, SHOULD give us the data that we need to store. But as is often the case, developers start with the database and build from there. But when you model the database first, you are designing the data to match a process model that exists only in your head, and I think that can be a mistake.
>
> — *Paul Ballard*

> I knew I would catch some flak for that, but let me explain. However, first, isn't your phrase: "Data only exists to store the state of some business process" the same as mine in that "processes only exist to manipulate data"? It's not the process we're after it's the data, the process is the means the data is the goal.
>
> WebServices and SOA offer nothing new here, they're still just data - they just might not be lying around in some DBMS (or at least served up that way).
>
> To me, the GEFN (Good Enough For Now) approach is far better, cheaper, and more realistic in that 5 years from now the applications we're creating will be on the trash heap of history or completely rewritten. I submit to you that the data will live on not the business objects, not the processes, not the UML diagrams, or even WebServices, they'll all come to pass. But the data, that will live on.
>
> I know you'll probably disagree with me and that's okay. Show me a data model and I can tell you what edit screens (web pages) and listing screens you'll need and what the overall application looks like, then I can simply generate the business objects from that meta data (using our product MyGeneration). Think about it, this is exactly what we do with Visual Studio, we point to a WSDL file and generate business objects that talk to that webservice, we do the same with database meta data.
>
> Programming languages will change, we just moved from ASP to ASPX, but many of us are using the same exact database tables and stored procedures that outlived the other technologies, and if I can point and generate both business objects and even UI code why bother planning on making my business objects live for 10 years? The simple truth is they won't, to much will change in the future to make it worth my effort.
>
> While most are hashing out diagrams and abstract domain models that often aren't really that useful my approach will be have me nearing completion, and with a better code base I think. Of course, these are just my experiences, I have no concrete study to provide you.
>
> — *Mike Griffin*

> As predicted, I disagree. :-)
>
> > isn't your phrase: "Data only exists to store the state of some business process" the same as mine in that "processes only exist to manipulate data"? It's not the process we're after it's the data, the process is the means the data is the goal."
>
> In a word, "no" this is not the same. Process is the goal, not for the developer perhaps but for the business. The company wants to notify customers of a new product, that is the process. The customer's address is the data that is used during that process. If we didn't want the process, we wouldn't need the data.
>
> Your statement that web services and SOA are nothing new, is a sign of how many developers don't understand the purpose of SOA. If all we wanted was universal access to data, we have that. Put it in Oracle, put it in SQL Server, heck even Sybase. There are very few technologies that can't access data in any reasonable commercial database. What SOA is trying to leverage is how I as an enterprise use that data, the process. So for example, if my company specializes in healthcare

> and we offer web services that allow you to track drug usage throughout a chain of hospitals, what would a pharmaceutical company want to leverage to track their product, our data or our process? The data they already have, just by looking at receipts. What they want to use is our business process of aggregating the data from the hospital chains. It's a fine distinction, but an important one.
>
> The data will live beyond the processes only because processes need to change more frequently than the data. How a company sends it's mail may change ten, even hundreds of times before ZIP codes become 10 digits instead of 5. But what is more likely is that the company will switch to sending correspondence via email, in which case the address data while still accurate is now irrelevant. Build to change, not to last. But with each new process inevitably comes new data. Most new systems built on old data structures perform miserably and don't take advantage of new technology.
>
> The ability to generate an object model or some amount of code is strictly a developer productivity tool. Too many times those productivity tools cause us to misuse a technology, i.e. Datasets in web services. We shouldn't let the tools dictate how we build systems. ”
>
> — *Paul Ballard*

The above discussion (lightly edited) is taken from [1].

References

[1] The ServerSide.Net: Grady Booch Fires Back at Software Factories. `http://www.theserverside.net/news/thread.tss?thread_id=30539`, Last accessed: 23/06/2011.

Exercise

You are implementing a library system. The library contains books each of which have a title, author, publisher and ISBN number (alphanumeric). The library may contain multiple copies of a particular book; however, each copy is barcoded with a unique reference number that is internally generated by another part of the system.

Users may borrow, return and renew library books. A book's loan period, and the total number of books that may be checked out differ depending on the type of user. The number of allowed renewals is four and is fixed for all users.

Users are fined if books are overdue. Fines may be paid at any time, but can only be paid on books which have been returned. Only the total fine may be paid and not a fraction of it. Books may only be checked out to a user provided the total accumulated fines and the fines on currently borrowed books are less than R 50-00.

The system will allow users to search for books based on the title, author, and other attributes.

Try the following tasks:

1. Give a UML diagram illustrating the classes required for modelling the above system. Only provide *key* attributes and/or operations for your classes. Think of your classes in terms of the building blocks of domain-driven design, and identify these on your diagram. Remember the building blocks are:
    - Entities
    - Value Objects
    - Services
    - Aggregates
    - Factories
    - Repositories
2. Give a sequence diagram showing the object interactions when a book is checked out of the library.

Exercise on Specification Pattern

```
class DelinquentInvoiceSpecification extends InvoiceSpecification
{
    private Date currentDate;

    public DelinquentInvoiceSpecification(Date currentDate) {
        this.currentDate = currentDate;
    }

    public boolean isSatisfiedBy(Invoice candidate) {
        int gracePeriod = candidate.customer().getPaymentGracePeriod();
        Date firmDeadline = DateUtility.addDaysToDate(candidate.dueDate,
            gracePeriod);
        return currentDate.after(firmDeadline);
    }

    public string asSQL() {
        return
            " SELECT * FROM INVOICE, CUSTOMER" +
            " WHERE INVOICE.CUST_ID = CUSTOMER.ID" +
            " AND INVOICE.DUE_DATE + CUSTOMER.GRACE_PERIOD" +
            " < " + SQLUtility.dateAsSQL(currentDate);
        }
}
```

In the InvoiceRepository class we want:

```
public Set selectSatisfying(InvoiceSpecification spec) {
   // give implementation
   // ...
}
```

Client code:

```
   Set delinquentInvoices = invoiceRepository.selectSatisfying(
       new DelinquentInvoiceSpecification(currentDate));
```