

Joint Application Design (JAD) – A Case Study

White Paper

Published: June 2002 (with revisions)

What You Don't Know Will Haunt You.

Contents

Preface	1
Introduction	1
Discovery	2
Assessment	3
Approach	3
The Kickoff	4
Obstacles and Concessions	5
The Goal is now in sight!	6
The Final Outcome	6
Summary	6
About the Author	6
Let Us Help You Succeed!	6

Preface

For many application or systems developers, the very first phase – gathering and defining requirements, is often the most daunting. The use of a proven structured approach can overcome both the actual and perceived difficulties, in establishing complete and concise requirements. Joint Application Development (JAD) offers a time-tested method for handling the business and administrative challenges in requirements gathering – not the theoretical, but the real world issues confronting developers. This White Paper will give you a revealing first-person look at the benefits of JAD.

Introduction

My first assignment as a consultant, many years ago, was to “create a database” for an insurance company that had just completed a design effort for a new Claim Processing / Risk Management system. This seemed like an easy enough project – something I could easily complete, and something that would make me look good for my new employer.

I was not given much information on this project, other than that the current system ran on a mainframe, and used a different database product. I was told the company spent the better part of the last year designing the new system, and the past 6 months designing the database. My thought was that this had to be a “slam dunk”. Not a bad way to start a new career.

What I found completely surprised me. There were technical issues such as configuring the UNIX system and installing the RDBMS product. No big deal. Then I was presented with a very large binder containing a textual design document of a schema with over 800 tables.

A review of the database design raised many questions. I asked to see the schema from the production (DB2) system. What I found were many similarities (probably 80%), but also some unique differences. I also found many incompatibilities between the schema design and the new database product (Ingres). The designers knew DB2, but did not understand the system they were migrating to.

There were datatype issues (for example, they specified the decimal datatype, which was not available at that time) and issues relating to the maximum row size (they specified 4 KB rows in many cases, but Ingres had a 2 KB row size limit at the time). There were also very many generic text fields that were intended to address “unknowns”. The problem with that is not getting the data into those columns, but rather maintaining and retrieving it in a quick, consistent manner.

Not quite what I expected for my first week. But, wanting to do the best job possible, I dug a little deeper...

Discovery

At first, I looked for a “road map” for the new schema, the Entity Relationship (E-R) Diagram. There was none to be found. Knowing this system would interface with various other systems, I decided to look for a Data Flow Diagram (DFD). There was a very high level document showing what happened conceptually, but it was not what I needed. So, I began to ask questions, creating this documentation as I moved along.

A simple question led to a big discovery. The new system needed to integrate three separate business units from two distinct companies: a commercial lines group; a personal lines group; and a company that was recently acquired, which did both. I asked, “How is a claim uniquely identified?” and got several very different answers with justification as to why their method was best. **Prior work on this project had assumed that “a claim is a claim is a claim,” which was not the case.**

Had this been identified later in the development effort it would have created costly rework and delays, and might have even been viewed as a “change” (which would have been incorrect). This was a huge finding, and at the right time.

At that time those three organizations did not share data. The changes I had found in the new schema (that 20%) were intended to handle this integration, but nobody every mapped the data between the existing systems and the new (proposed) schema. It was clear that there were problems, and that simply moving the data to a central repository would not provide the desired business results.

The logical progression of this was that I asked about business rules. There were many assumptions about the business, both on my part and on the part of the various people who had been working on this project for so long. There was knowledge about the production application systems from the mainframe personal lines claim system. **Overall were bits of knowledge woven together with threads of assumptions.** Up until now this had not been viewed as a big priority. The mindset seemed to be that the processes would somehow fit the new schema.

Assessment

I performed a high-level gap analysis of where I felt this project was; relative to where I felt it needed to be. I contrasted that assessment, with the client's assessment of where they were. There was a big difference of opinion.

Development was scheduled to begin. The first prototype was due in 6 months, and the initial system rollout was schedule for 1 year out. Making any changes at this late stage was risky, and would be a hard sell. But, I knew if the changes were not made, the product would not meet expectations.

This was the first implementation of the new database product at this company. My fear was that if the project failed, the database product would be blamed. Since I worked for that vendor, and was the first consultant on the project, it seemed this failure would be attached to me personally. That was certainly not what I wanted for my first engagement. And this was only the middle of the second week!

Approach

It seemed if I was really going to do a good job I would need to resolve the issues that had been identified so far. I began developing a technical case to justify my assertions. I then built a business case to justify making changes this late in the game. Luckily, I reported to a very astute Director who understood what I was telling him and believed me. He understood the time constraints and urgency, and asked me what we could do.

During the first seven years of my career, I had had the good fortune to design, develop, and implement, several new application systems. The systems were all successful, but the design effort was generally very time-consuming. Given the perceived probability of failure it seemed like drastic measures were required.

I had read an article about an interesting approach to system design using something called “Joint Application Design” (JAD). Having worked earlier in my career on “Rapid Application Development” (RAD) projects that had varying levels of success, I immediately saw the value of this process. It seemed like the exact tool to quickly identify and work through the issues that had been identified, and to start building consensus about how the new system would look and act.

I had never actually performed a JAD session, and was very up-front about that fact. The Director liked the idea. He arranged to have the JAD sessions begin the following Monday. There were a total of around 30 people, identified to represent the 3 functional areas. The goal was to have a working schema by the end of the week. Good thing they purchased 30 days of consulting!

The Kickoff

The first day of the JAD session was very interesting. The common perception was that the proper analysis had been performed, and this was just a waste of their time.

Having had a lot of experience with data modeling, I decided to take a risk to make my point. A “claim” was a core component of their new system. From what I had seen so far there was no clear definition of a claim, nor was there documentation on the business rules and exception processing. So, I created a table diagram on the white board that had “claim number” – nothing else. I asked someone for an example of a claim number so we could populate this simple data model with representative sample data. Guess what? There were three different methods of uniquely identifying a claim. One was purely numeric, while the other two were alphanumeric, but had different properties.

Luck was on my side and I made my point. People now saw the value of this exercise. And, not having a lot of industry experience allowed me to ask “dumb questions” that often opened the door to new issues and requirements. Over the course of the first day it became clear to the group that the “dumb questions” were not only good question but also required given the differences between the environments. We very quickly worked through the various assumptions and coming to a common understanding of many aspects of their business.

I had assigned a “scribe” to document the discussions, agreements made, issues raised, and positions taken by various people. Each day generated 20+ pages of notes. At the end of each day, I reviewed those notes, added my personal commentary, and generally made sure the essence of the issues and agreement was captured.

My experience with designing new systems and database schemas was that a three-schema approach generally worked best. That process is described on the next page.

The first schema is the *Business Schema*. It captures the business requirements of the new system. The schema was validated by “working” representative sample data through it. This was a good way to identify business rules and exception processing. This also allowed us to create an E-R Diagram and Data Flow Diagram “on the fly”.

The second schema is the *Logical Schema*. This is where we normalize the database, making it the theoretically best it can be*. The sample data is transformed, and again worked through the schema. Once you can prove there was no loss of functionality, it is time to move on.

The third schema is the *Physical Schema*. This is the practical schema ensuring data integrity, while promoting good performance.

*Aside: My very first database design was a wonderfully normalized database. Third and Fourth normal form everywhere. No dependencies here. I was especially proud of a query that took up more than full page (60 line greenbar)! The first time I ran the “perfect query” in the “perfect database”, it took one full day to complete. Over the course of a week I tried several things. Eventually, I carefully de-normalized the schema. Sure enough, the much simpler query ran in a few minutes. This was a very valuable lesson I was lucky enough to learn early in my career. End Aside.

By asking the right questions, using representative sample data to “work” the model, and engaging the team, we were able to make incredible progress that first week. I truly believe we accomplished more in that first week than had been accomplished during the analysis and design efforts from the preceding year.

Obstacles and Concessions

Everything was not perfect though. There were semantic issues we could not resolve, such as the difference between a policy and a service agreement. While functionally similar, I was unable to convince the group to have a combined policy and service agreement table. What I was able to get, was agreement that subordinate entities could be shared.

Other concessions included things like being able to maintain the native claim identifier. I was able to convince the team to use a surrogate key as a primary identifier, and to use a free form text field to capture the original claim identifier. By capturing the functional area, I could design a schema that provided uniqueness on the original identifier, while still maintaining uniqueness throughout the table using the new surrogate key. A later side benefit of this was that we could physically cluster the data by business unit, which improved both performance and concurrency.

But in the end we had something I viewed as being less than perfect but still very good. It achieved the business goals and objectives established for the JAD session and provided documentation and a common understanding of all systems involved. Best of all, it was something we could actually implement!

The Goal is now in sight!

The weekend following the first week of the JAD session, I completed the data modeling effort. I populated the logical schema with our sample data and developed various test queries, views, and simple reports. These were presented to the team members at the beginning of the second week.

We spent two days refining and tweaking the schema and business rules. At the end of that week I presented a report containing: a summary of the activities performed; a summary of the various discussions / issues / action items; as well as issues and position statements for the various business units. There were still minor issues to be addressed, which took about 3 months to completely resolve, but the general consensus was that we were very successful.

Houston, we have lift-off...

The Final Outcome

The prototype system performed all required functionality. Data migration efforts were successful. Performance issues we identified and corrected. The project was a little behind schedule, and slightly de-scoped, but it did go live on the original target date. The project was a success, and my short project had extended to over 1 year in duration.

Summary

Was it worth it? Yes. Would the system have been successful had we not performed the JAD session? Doubtful. Was I sold on JAD? Absolutely. When done right, JAD saves time and money!

About the Author

Chip Nickolett, MBA, PMP is the President and Founder of Comprehensive Solutions. He was a Senior Consultant with the Ingres Products Group of ASK when he performed this project. Since this project, he has successfully used JAD techniques to expedite projects of various size and complexity.

Let Us Help You Succeed!

Call today to discuss ways that Comprehensive Solutions can help your organization save money and achieve better results with your IT projects. We provide the *confidence* that you want and deliver the *results* that you need.

[View our "Confidence" Brochure](#)

[Back to White Papers](#)

[Back to Services](#)

Comprehensive Solutions
4040 N. Calhoun Road
Suite 105
Brookfield, WI 53005
U.S.A.

Phone: (262) 544-9954
Fax: (262) 544-1236

Copyright © 2000-2008 Comprehensive Consulting Solutions, Inc.
All Rights Reserved

No part of this document may be copied without the express written permission of Comprehensive Consulting Solutions, Inc., 4040 N. Calhoun Rd., Suite 105, Brookfield, WI 53005.

This document is provided for informational purposes only, and the information herein is subject to change without notice. Please report any errors herein to Comprehensive Consulting Solutions. Comprehensive Consulting Solutions, Inc. does not provide any warranties covering and specifically disclaims any liability in connection with this document.

All product names referenced herein are trademarks of their respective companies. Use of these names should not be regarded as affecting the validity of any trademark or service mark.