

Global Electronics Corporation
End-to-End Quality Monitoring System

Agility Workshop and 30-Day Blitz

Project Team Report

Blitz Results and Lessons Learned

Prepared by:

Center for Systems Innovation



Table of Contents

The 30-Day Blitz	Page
IT Agility in Action	1
Six Techniques Lie at the Heart of the 30-Day Blitz	3
Composition of 30-Day Blitz Team	5
The Define – Design – Build Process	6
Benefits of the 30-Day Blitz	9
Managing the Encounter with Complexity	10

Tracking The First 30-Day Blitz

Define - 2 Days - May 7-8	11
Design - 6 Days - May 9-16	12
Build - 11 Days – May 17 – June 1	15

Company Context and Lessons Learned

War Room	20
Sporadic Involvement	20
Pace of Brainstorming	21
Business Action Framework	22
Letting Go of Perfection	23
Roles	24
Object Model	24
Overlapping Design and Build	25

BLITZ Team Members

Paul Keane (System Builder)	Cynthia Ockleberry
James Dennis	Jim Duncan
Peter Orenberg	Tim Brooks
Terry King	Venkat Prasad
Michael Hugos (Coach/CSI)	Carl Pregozen (Coach/CSI)

The 30-Day Blitz

IT Agility in Action

In every situation there are significant improvements that can be made in 30 days or less; and they become a base to continue building further capabilities as you move toward your goal. It is very important to jump-start a project by finding an opportunity to deliver something with tangible business value in the first 30 days. This is called the 30-Day Blitz.

Projects need to start off with a quick delivery of something the business people can immediately put to use. This is so important because it provides proof that the project is headed in the right direction and that it will live up to expectations (at least most of them). People are very busy these days, and they are wary of system development projects because they have seen so many of them fail. They need to see something positive happen quickly before they will really commit their time to a project.

Delivering the “Robust 80% Solution” and Building Momentum

Here’s a quick story to illustrate this notion. Some years ago I (Mike Hugos) was the project leader on a project for a financial services company that provided financial reporting to firms that traded in futures of various commodities such as agricultural products, fuel oil, and other raw materials. The company provided trading data to the accounting departments of the commodity trading firms; it showed the firms the trading activities of their commodity traders.

Since it was possible for a firm’s traders to do thousands of trades very quickly, the accountants felt the need to get better intra-day trading data so that they could better track the risks their traders were taking. The financial services company decided to invest in developing a multi-million dollar state-of-the-art system to provide this trading data in a streaming, real-time environment.

But, as often happens with such projects, the work got bogged down with expensive and complicated technology that did not work as expected. I was asked to reorganize and reenergize the project. I plunged in and began to investigate the project team and the state of the system that had been created so far. I found a lot of demoralized programmers and very skeptical business users.

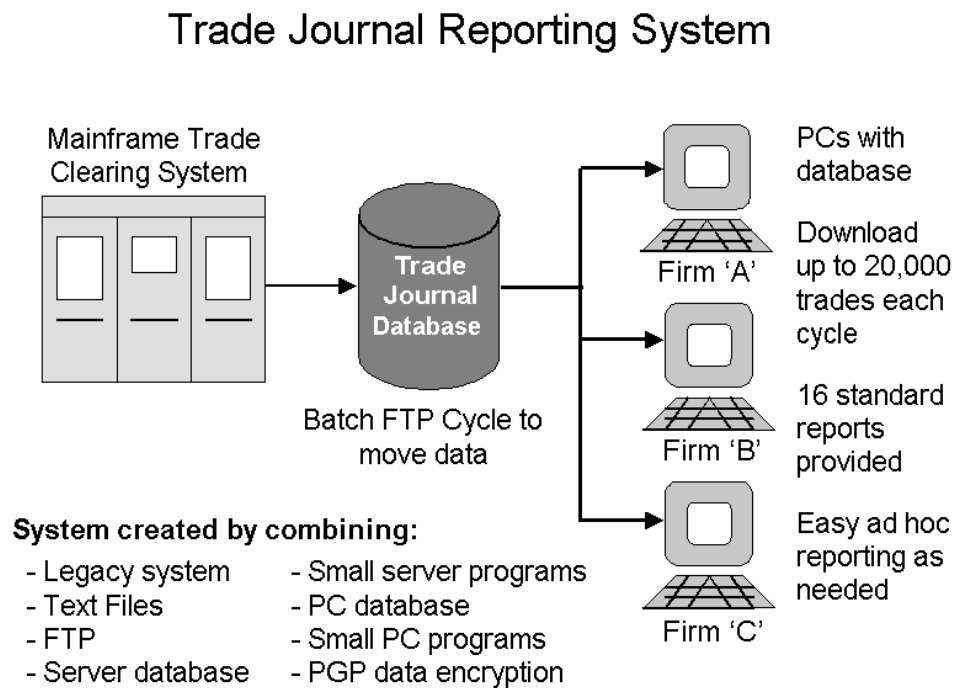
The project needed the active support and participation of all the stakeholders if it was going to stand a chance of succeeding; yet, people were reluctant to give this

commitment for obvious reasons. The project had been underway for a bit more than a year and had so far failed to meet expectations. Who wants to commit to a failing project?

While all the heavy lifting was going on with trying to reorganize and turn around this big project I was able to get permission and a little funding for a quick project that would leverage existing systems and require just a few weeks of programming to put into production. This project had the potential to quickly deliver many of the most important trading reports that were being requested by the trading firms. It would not do everything the big system was supposed to do, but it would start providing customers with something very valuable that they could use until the big system was finished.

See Figure 1.1 for a conceptual design of this system. The system was in production in three weeks. It was built by pulling data from some existing systems and combining readily available IT components that all the business people already had; the components were office productivity tools like spreadsheets and personal databases. These components were connected together by small chunks of program code that we wrote to move data between them and do calculations needed for a basic set of trading reports.

Figure 1.1



You may be thinking, “Well, gee, that’s awfully simple.” Yes, it is. That’s why it’s so powerful. The system went into use fast, and within several weeks we were working on adding the next round of enhancements to it. We were improving the data collection process; we were adding more data security and additional error checking. There was talk of a whole new set of features to do performance monitoring and alerts too. The people using the system put together a list of other features they wanted.

What It All Might Mean

First of all, business people do not care about technology, they just want the data and the capabilities they need delivered to them as quickly and simply as possible. Second, keep scope narrow by delivering the 20% of capabilities that provide 80% of the value (the “Robust 80% Solution”). Third, keep IT super simple (KISS); it’s often better to build systems where the user interface is composed of common office productivity tools like spreadsheets, web browsers, word processing, and personal databases because people already know how to use these tools so there is much less of a learning curve.

There is also an empowering, galvanizing shift in perspective that comes from this approach. By focusing on high value items to that can be delivered within 30 days, the mindset of the team is moved away from figuring out how to explain why success can’t be had. Instead, they have to ask themselves, “What *can* we accomplish?” Language defines reality: Here, a powerful, reinforcing pattern of accomplishment is established in place of such possible unproductive behaviors as analysis paralysis, self-pity, and rationalization.

This is agile IT. It delivers major business benefits very quickly. It is a fast, low-risk way to jump-start a project. You don’t even have to say much about it until you deliver the first big win. (That way you don’t needlessly raise expectations.) And, once you do deliver that first big win – that “robust 80% solution” – then you have the credibility and momentum you need to keep right on going.

Six Techniques Lie at the Heart of the 30-Day Blitz

Your skill in using just six techniques in appropriate combinations to address different situations is the key to success in a 30-Day Blitz. I call these the core techniques. Just as the game of basketball has a small group of basic techniques (such as dribbling, passing, guarding and shooting), so too does the game of developing information systems.

The skill levels of project teams can be measured by their capabilities in these techniques. By employing these techniques, a project team will always be able to produce competent (and sometimes even brilliant) results. These six techniques are a simple yet comprehensive set of skills that can be taught and mastered by people involved in building systems. They are:

- 1) Joint Application Design (JAD)
- 2) Process Mapping
- 3) Data Modeling
- 4) System Prototyping
- 5) Object Oriented Design and Programming
- 6) System Testing and Rollout.

Other techniques may be relevant from time to time, but these core techniques are always relevant in every situation regardless of the technology being used or the problems being addressed. The best practitioners of the 30-Day Blitz, and IT agility in general, are competent in all six techniques and masters of some of them.

These techniques are well known in the IT profession; they weren't created yesterday. Do a web search on any of these techniques and you will find hundreds of references to them and many options for getting education and training in them. In the game of IT agility, these are the basic skills of the game. It is possible to get a very accurate assessment of the chances for success of a project team on an agile development project (or any IT development project) by measuring their skill levels in these six core techniques.

Of course people on project teams need to be skilled in the use of the specific programming languages, operating systems, databases and software packages that will be used, but without solid competency in the core techniques, they cannot be agile. Without competency in these techniques, whatever technologies people use tend to be employed in complex and clumsy ways. This is because people who are not competent in the core techniques tend to learn particular technologies by rote memorization, which limits their options for flexibility and responsiveness.

In a fast-paced, agile organization, people need higher skill levels in these core techniques than what was required in the "good old days" when companies and technologies did not change so fast. In those days people could spend years learning a particular programming language and they could spend years learning the operating procedures of their organization. Rote learning of technical details had a chance to produce value because things stayed the same long enough for the rote learner to come

to an understanding of higher level requirements by digesting masses of low level details.

These days as things change faster and faster, it is critical for people to be competent in the use of higher level skills that allow them to analyze many different situations, organize and communicate their ideas, and design solutions to new problems using many different technologies. People with these skills will remain in constant demand regardless of the technologies being used and the organizations with whom they work.

IT agility happens when people use creative combinations of these techniques to define, design and build solution systems for the challenges they face. The JAD technique is a key technique for pooling the insights and ideas of business and technical people to define possible solution systems. Combinations of JAD, process mapping, and system prototyping are used to further explore possible solution systems and create detailed system design specifications. Building working systems from design specifications calls for the techniques of object oriented design and programming and system test and rollout.

Composition of 30-Day Blitz Team

A Blitz team is made up of three to nine people. Teams always contain a team leader or “System Builder”, a senior business analyst, and a senior developer; additional team members are added as needed:

Team Leader/System Builder – 10+ years experience. Skilled in all six core techniques.

Sr. Business Analyst – 10+ years experience. Skilled in five of the core techniques: facilitation of joint application design sessions; workflow process mapping; logical data modeling; user interface design; system testing and rollout.

Sr. Developer – 10+ years experience. Skilled in four of the core techniques: data modeling; system prototyping; object oriented design and programming; and system testing and rollout.

Business Analyst – 3+ years experience. Competent in some of the core techniques used by a Sr. Business Analyst.

Developer – 3+ years experience. Competent in some of the core techniques used by a Sr. Developer.

Sr. Data Modeler/Database Analyst – 6+ years experience. Skilled in logical and physical data modeling, programming of stored procedures, and optimizing performance of databases as needed for rollout of new application systems.

Sr. Network Support Specialist – 6+ years experience. Skilled in deployment of hardware, operating systems, and telecom connections needed for roll out of new application systems.

Sr. Project Manager – 10+ years experience. Skilled in the use of project management techniques appropriate to support the fast-paced work rhythms of the 30-Day Blitz. Maintains updated project plans and budget showing time to complete and cost to complete for the systems each blitz team is developing.

On larger projects there are several project teams working in parallel and their actions are guided and coordinated by a senior team leader called the “System Builder”. When there is only a single team on a 30-Day Blitz then the system builder becomes the project team leader.

The Define – Design – Build Process

A 30-Day Blitz is structured by using the Define – Design – Build process. This process provides a framework to guide the system builder in the selection and use of a combination of core techniques that are appropriate to any specific situation. Define – Design – Build is a set of tactics or “ways to get things done” based on using different combinations of the six core techniques. (See Figure 1.2.)

The power of Define – Design – Build is that it is based on the use of this small yet comprehensive set of six core techniques. It is possible for everyone on a system development team to understand and use them effectively. The result is that systems get built in a very timely and efficient manner. The system builder can combine the six core techniques in an almost endless number of ways to meet the demands of each specific situation.

Define – What to Do

In the Define step the system builder uses JAD and process mapping to work with business managers who will sponsor the project. Business management defines the business goal and the performance criteria they want to meet. The system builder and other team members create a conceptual design for a system that will achieve the

performance criteria laid out by business management. The conceptual design literally is the strategy that will be used to accomplish the business goal. The project objectives are then to build the various system components laid out in the conceptual design.

A cost benefit analysis is done to evaluate whether the cost of building this system is justified by the benefits it will provide. If the costs are too high, then a different conceptual design is created that still meets most of the performance criteria and is less expensive. If a conceptual design can be created where the costs are justified by the benefits, then the initial project plan is put in place, and the project moves into the Design phase.

Design – How to Do It

In the Design phase, the system builder directs the activities of a design group that creates the detailed system design. In this phase they build a detailed prototype of the system's user interface. People who will have to use the system get to "test drive" it. The group also builds a technical prototype composed of the hardware, software, and operating system specified by the system design. This verifies whether the technical components of the proposed system will work together as claimed and whether they will meet the performance levels desired.

The system design can be adjusted as needed to produce a system that will perform effectively. Changes in the user interface and the technical components of the system can be made based on test results. When the design is complete, an updated project plan and budget can be put in place to guide the work in the build phase. Also, there is the option to cancel the project without a big loss of capital if the design phase shows that the system cannot be built to perform as expected at a reasonable cost.

Build – Just Do It

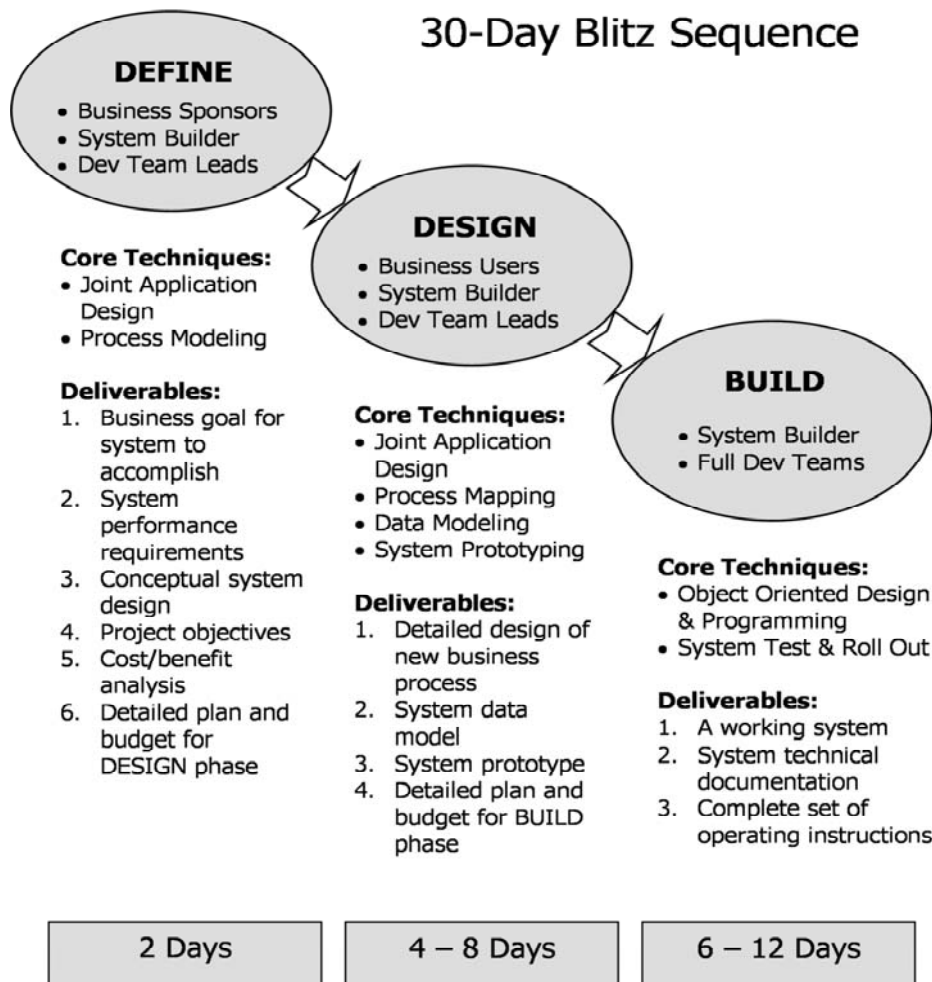
In the Build phase, the full complement of people are brought onto the project, and the various teams on the project focus on building the system components assigned to them – their project objectives. There should be no further design questions to answer, and everyone from the system builder to the team leaders and the individual team members can turn their full attention and energy to the job of getting the system built.

Define – Design – Build is a process that uses time boxing to provide maximum traction and propel the project along. As the saying goes, "The job will expand to fill the time available." Therefore we respond to that tendency by time-boxing our activities. When

someone asks, “How do we know that the Define phase of a 30-Day Blitz will take only two days,” we answer that it will take that long because that is how much time we have allocated. That time should be sufficient for the situation confronting us. We will use selected core techniques to do a competent job, and at the end of two days we will know what we know. In light of what we know at that time, we will either proceed into the Design phase or cancel the project.

If you adhere to the time-boxing guidelines, you will generate forward momentum on the project very quickly. The challenge for the project team becomes to use the relevant core techniques competently within the time allocated to get various tasks done. The challenge for the system builder is to allocate time and resources effectively. The system builder must exercise the skills of designing systems and leading projects to ensure that the project stays on track.

Figure 1.2



These are demanding challenges for everyone involved. Yet is also exhilarating for everyone involved to participate on a project where people rise to these challenges and things get done so quickly and effectively.

Benefits of the 30-Day Blitz

From the perspective of the senior business executives who sponsor a system development project, the 30-Day Blitz and the Define – Design – Build process is a way to manage project risk. In the Define phase small amounts of time and money are spent up front to qualify a business opportunity. If findings warrant, the company then spends only a moderate amount of further time and money in Design. During Design, a small, prototype system is created to prove that the opportunity is real and justifies a larger investment. The Build phase is where the bulk of the time and money is spent; yet, the decision to move into Build is made with the greatest amount of information. The nature of the business opportunity and the solution system that will exploit that opportunity are well established.

From the perspective of the system builder, the Define – Design – Build process is a way to navigate through the 30-Day Blitz and the complexity of creating a new system. The system builder is truly the person on the hot seat who needs to get things done. The Define – Design – Build process provides a framework to structure the work sequence. It lets the system builder set reasonable time limits within which to investigate situations and make decisions in the Define and Design phases. When decisions about system design and budget have been made, this same framework provides a set of tactics for the system builder and the team leaders to employ during the Build phase. The core techniques employed in the Build phase give a lot of structure to the work and enable the system builder to effectively lead the effort.

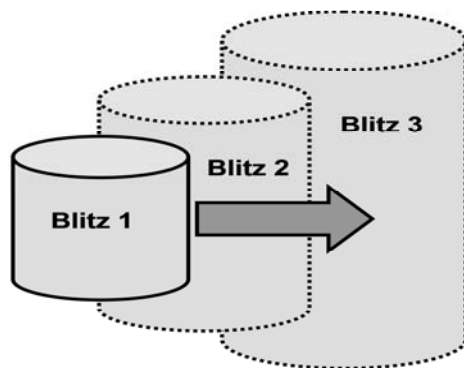
From the perspective of the people on the project team Define – Design – Build is a clearly defined and manageable repertoire of techniques to use. People who participate in the project in each of the three phases know which of the core techniques they will be expected to use in the conduct of their work. They can focus on mastering these techniques. And there is also an emphasis on working together on tasks in small groups so that the learning and use of techniques among team members is more effective than if they worked alone.

Managing the Encounter with Complexity

Define – Design – Build is a three-step sequence that could also be described as, “Move it! Move it! Move it!” In order to support an agile corporation in a real-time world, systems must be built in a quick and iterative process. The time boxes for each step in this sequence are to be strictly adhered to. A disciplined and fast-paced approach is the best way to handle complexity. Complexity is as much perception as it is reality. If you allow yourself or your project teams too much time to contemplate the situations facing them they will tend to perceive excessive amounts of complexity and sink into that dreaded state called “analysis paralysis.”

Business advantage belongs to those who learn to deal well with complexity. Business initiative goes to those who learn a process that guides them in defining business opportunities and then exploiting those opportunities in a fluid and coordinated manner. One opportunity often leads to the next. Respond to each opportunity with iterative 30-Day Blitzes and teams of people trained to use the Define – Design – Build process. Define opportunities, design systems to exploit those opportunities, and quickly build those systems. As success with one opportunity opens up other opportunities, continue to respond using the Define – Design – Build process. Your organization will build great competitive momentum this way.

“Think big; start small; deliver quickly!”



Tracking The First 30-Day Blitz

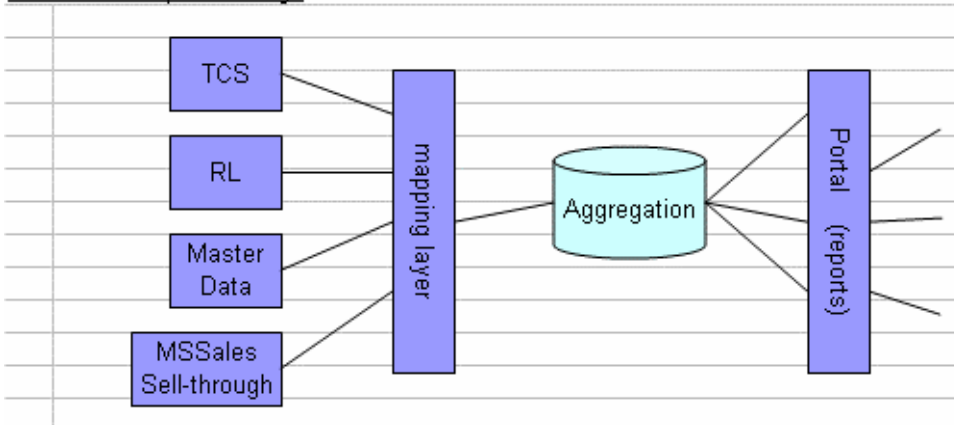
Define - 2 Days - May 7-8

Agility Workshop		Sequential Biz Day, Cal Day
Dt	Work Accomplished	
	<ul style="list-style-type: none"> ➤ Defined / began 30-Day Blitz approach ➤ Business intro to symptoms & problems: data, process, outsourced participants, misaligned motivation 	1,1
5/7	<ul style="list-style-type: none"> ➤ Defined business goal: <i>Develop end-to-end quality system by integrating data across the business to enable improvement of customer satisfaction and product quality.</i> ➤ Defined initial objectives 	
	<ul style="list-style-type: none"> ➤ Produced visual supply chain map identifying data entities and integration trouble-spots 	2,2



- Revisited / refined bottleneck defs
- Refined system goal
- Defined initial **“test” questions** a system should answer
- Defined initial **conceptual model**

MIDAS Conceptual Design



Design - 6 Days - May 9-16

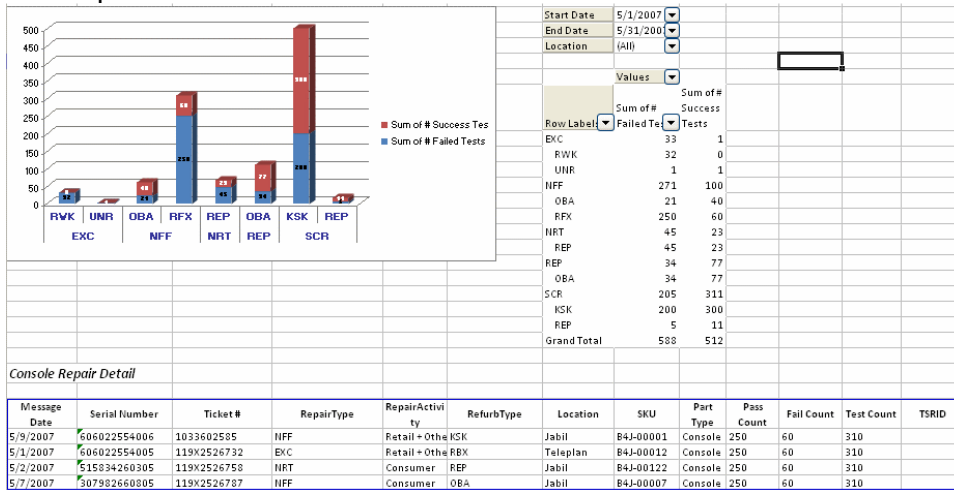
(Some DM items actually overlap Build, up to May 18.)
 JADs & Validation

Dt

Work Accomplished

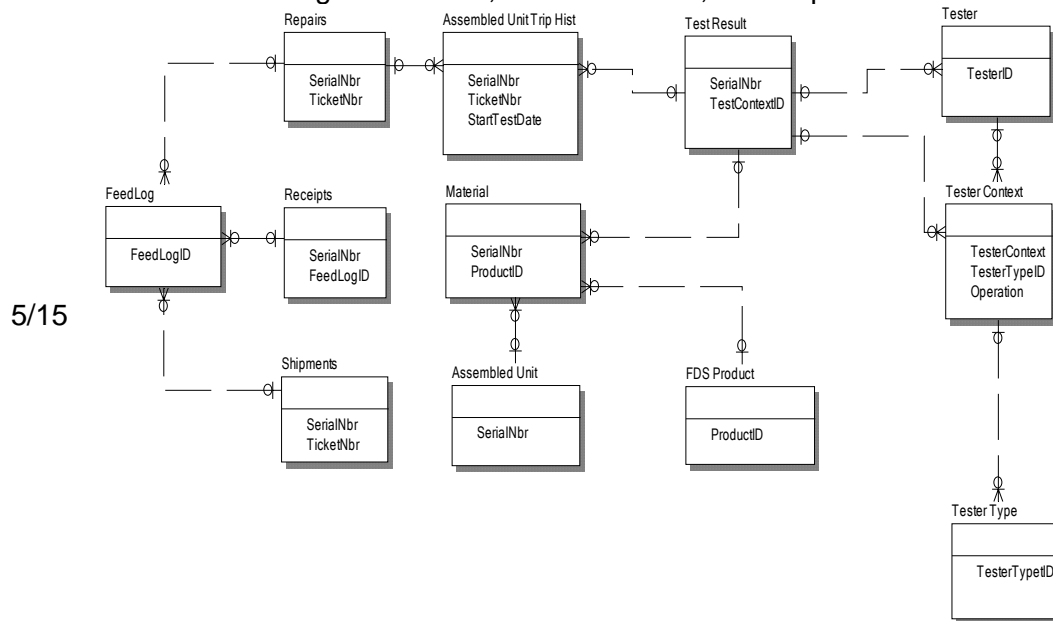
Sequential
 Biz Day,
 Cal Day

- Pre-test report w/ business:
 - ☆ Determined additional requirement – XPM link for component timing
 - ☆ Require TSRID field for hooks to TCS



- JAD #2
 - ☆ Confirmed high-level LDM, validation rules, and UI plan

7,9



5/15

- ☆ Business resource commitments?
- Formalized Design of **technical architecture**

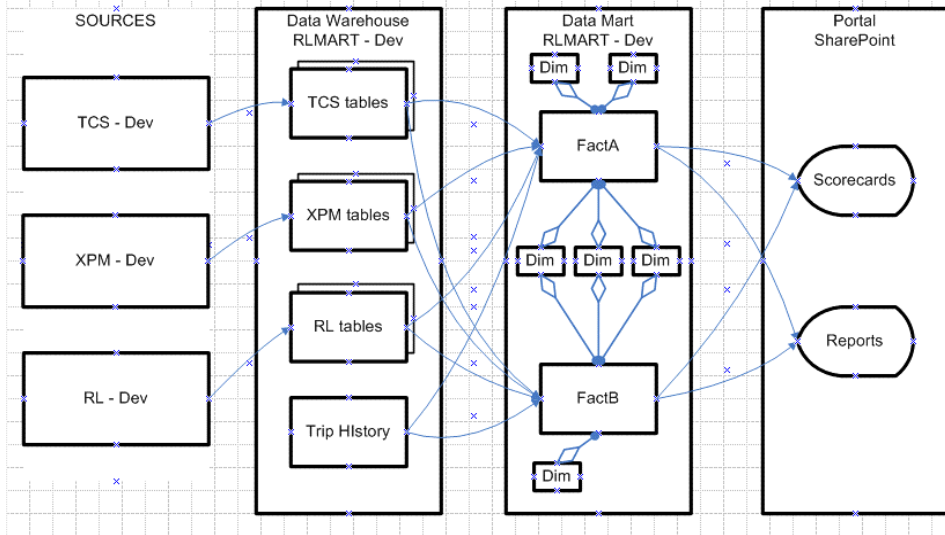
Design - 6 Days - May 9-16

(Some DM items actually overlap Build, up to May 18.)
 JADs & Validation

Dt

Work Accomplished

Sequential
 Biz Day,
 Cal Day

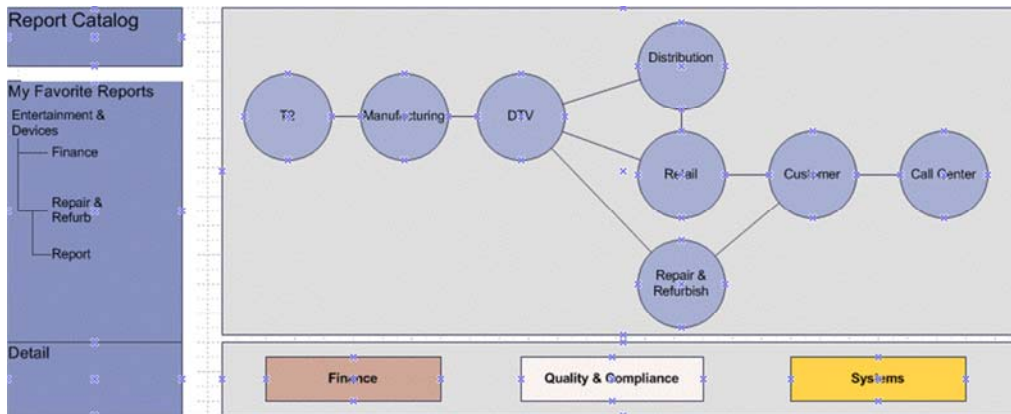


➤ Began DW build activities

➤ Formalized **schema of UI** screens and drill-downs

8,10

5/16



➤ Began testing technical architecture

Build - 11 Days – May 17 – June 1

(Some DW items actually overlap Design, as early as May 10.)
Implementation

Dt	Work Accomplished	Sequent Biz Day, Cal Day
----	-------------------	--------------------------



Project Team in war room, Building A

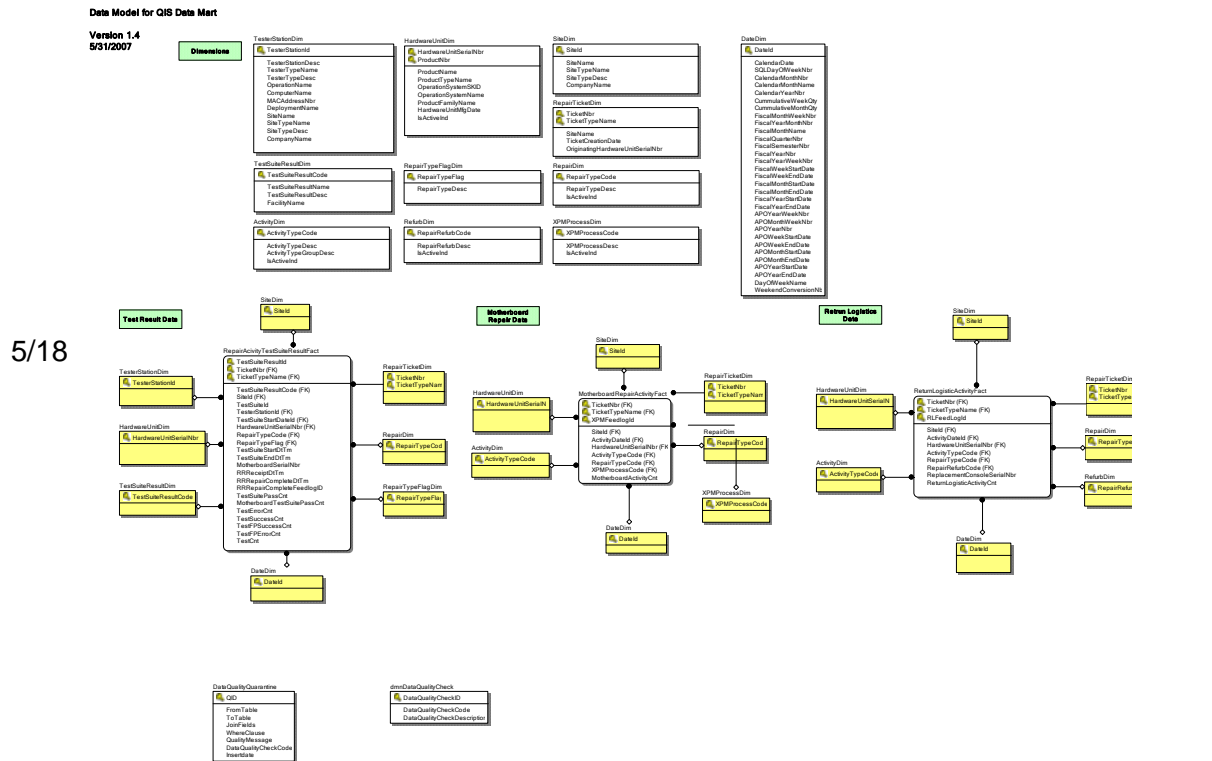
9,11

5/17



- Constantly added more detail and accuracy to project plan as we moved through the project
- Regularly analyzed project. plans for patterns, to avert common “gotchas”
- Continuously adjusted work and resources to meet finish date of June 1

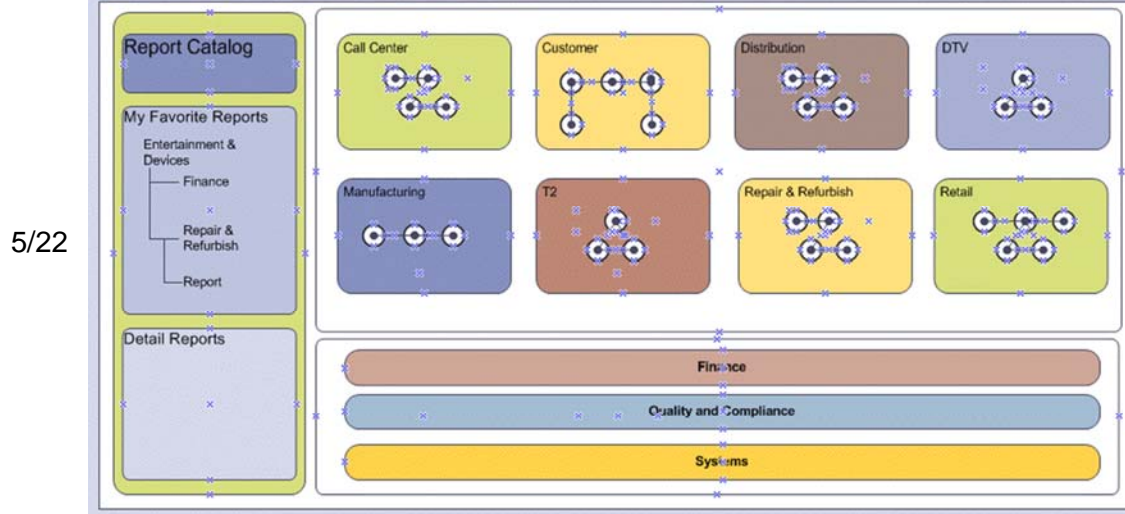
➤ Finalized design of Data Mart (dependent on Data Warehouse) 10,12

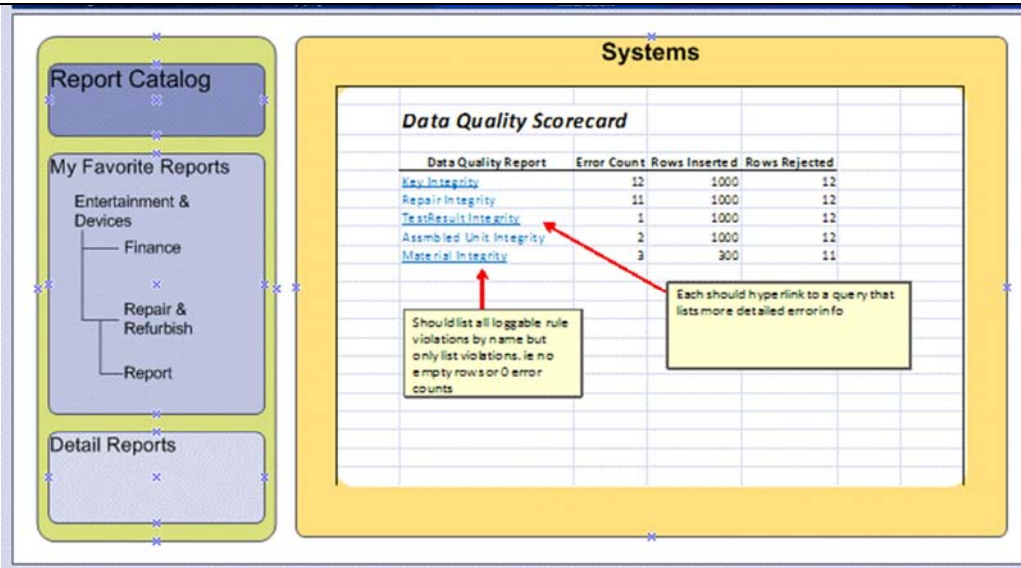


weekend

- 5/21 ➤ Created Data Mart DDL 11,15
- Created Data Mart Report Layouts

➤ Further UI Definition 12,16





5/23 ➤ Created DW Capacity Plan 13,17

5/25 ➤ Created DM Data Quality Checker 15,19

weekend

➤ Implemented E2E Test Plan 16,23

5/29

	WED 5/30	THU 5/31	FRI 6/1	PS ITEMS
	PASS 1	PASS 2	PASS 3	
	PASS	FAIL	PASS	FAIL
DMC DW LOAD	FAIL	PASS		36126
APL INTEGRITY CHECK	PASS	PASS		
DMC DM LOAD	FAIL	FAIL		36127, 36128, 36134 (Cubic Item)
B.S. R REFSH	PASS	PASS		
B.S. CUBE REFRESH	PASS	PASS		
V.P. SHAREPOINT: RENDER REPORT	FAIL	FAIL	PASS	
V.P. SHAREPOINT: NAVIGATION	FAIL	FAIL		
B.S. REPORT: DATA REC	PASS	PASS		
T.B. REPORT: UAT		FAIL		36124
T.R. FIX CUBE: UAT	FAIL	FAIL		
DMC DMC DRP ALL & PERW	FAIL	FAIL	FAIL	36139 C Uwg Idr NC Idc

5/30 ➤ Added Data Mart indices for performance 17,24

5/31 ➤ Refined data quality rules 18,25
 ➤ Tuned indices on Data Mart

➤ Cube and analytics set-up 19,26
 ➤ Finalized SharePoint webpage layouts
 ➤ Completion of testing

➤ **Kudos for a job well done – on time and quickly delivering value:**

From: Robert [redacted]
Sent: Friday, June 01, 2007 4:34 PM
To: Cynthia [redacted], Harjinder [redacted], McGinnis [redacted]
Cc: Carl Pas [redacted], exchange.com, Carl Tegezen (Enclave Systems Inc)
Subject: RE: MIDAS Release 1.0 - Weekly Status Report

This is incredible work to deliver a Version 1.0 solution in record time. Congratulations to the entire team on making this happen. Well Done!

6/1

Project Summary				
<ul style="list-style-type: none"> MIDAS will provide an integrated, scalable, and flexible quality management infrastructure that enables complete end-to-end visibility into product quality for predictability and decision-making across all of our businesses. MIDAS Release 1.0 (30-Day Blitz from May 7 - June 1) will build infrastructure to link data from Test Control System (TCS), Reverse Logistics (RL) and parts of XPM. Overall Percent Complete (Define, Design & Build): 100% MIDAS Team Site MIDAS Release 1.0 Project Plan 				
Overall	Schedule	Budget	Issues	Risks
↑	↑	↑	↑	↑

Milestone Summary			
	Start	End	% Complete
Define	5/7/2007	5/8/2007	100%
Design			
Data Warehouse	5/9/2007	5/18/2007	100%
Data Mart	5/15/2007	5/24/2007	100%
SharePoint	5/22/2007	5/24/2007	100%
Build			
Data Warehouse	5/18/2007	5/25/2007	100%
Data Mart	5/21/2007	5/29/2007	100%
SharePoint	5/29/2007	5/31/2007	100%
Test			
Data Warehouse	5/29/2007	6/1/2007	100%
Data Mart	5/25/2007	5/31/2007	100%
SharePoint	6/1/2007	6/1/2007	100%
Release 1.0 – Stakeholder Review	6/5/2007	6/5/2007	0%
Release 2.0 – JAD Session	6/7/2007	6/7/2007	0%

weekend

Company Context and Lessons Learned

War room – balancing collaboration and distraction

The project team has to be co-located in order to enable the kind of frequent, rapid, immediate communication that is required for IT agility. Furthermore, they need to be disconnected from the day-to-day distractions and interruptions they will encounter if they stay in their traditional environments. Therefore, a dedicated conference room – a “war room” – should be allocated for the team to use during the 30-Day Blitz.

During the Build phase, however, there are a few instances when multi-hour periods of intensive concentration are required by individual team members. At these times, the war room environment becomes burdensome, and the team members should be permitted to break off to their private worksites. This type of independence period should last no more than a few days, with participation in stand-up meetings still required (conference call is an option). When integration, testing, tuning, and debugging take place, the team must reconvene. It is up to the system builder to determine when the team should work in the war room and when they should be permitted to separate to their offices or other private areas.

Sporadic involvement – difficulty of on-boarding loosely-coupled resources

On a 30-Day Blitz, having a team member jump into action after the Design or Build phases are already in full swing is difficult. The team has momentum and doesn't want to slow down; yet the unfamiliar teammate needs information and patience. Imagine a person trying to join an in-process football play. They might get in the way of a receiver's running pattern or block the field of view of the quarterback. They don't have bearings and will struggle to get into the pace and mindshare of the team.

On longer Define-Design-Build projects, it is possible to bring in new team member as the team progresses into the build phase. In short 30-Day Blitz projects, the team doesn't have tasks allocated for bringing someone up to speed. Every effort should be made to identify all needed project team members before the start of Design. If the particular skills of a team member are not needed until later in the Design or Build phases, it is important to find a way to involve that team member early in the process (potentially with an interim alternate task) so that he/she becomes connected with the project and with the rest of the project team. If no alternative can be found but to start a

team member in the middle of the project, then put specific time for other team members in the project plan to account for on-boarding information sharing.

Pace of brainstorming – closing the loop on brainstorms

In the two-day Define phase, we found that some participants representing different parts of the business were highly vocal, and others were fairly quiet. Also, some were full of ideas while others' comments were more explanatory or self-protective. A significant and important activity in the Define phase is brainstorming – generating ideas and insights around business processes, needs, and bottlenecks. To be effective, this work requires an open, trusting atmosphere.

- In the context of “Define”, the group chose a problem definition and solution design that resulted in definition of the problem in a more technical and system specific manner.
- The broader business context for this particular system was not at first clear
- It was decided that the broad business context was the notion of “Supply Chain Visibility” – if all parties in the Xbox supply chain can see activity and performance metrics at each stage of the supply chain that will be what enables people to take timely and effective action to eliminate problems and increase through put.

In JAD session #1, the same team reconvened and the JAD session self-corrected – more narrowly defining the problem and refining the evaluation criteria. From this clarified problem definition, the project team was able to advance an initial system design that the stakeholders ratified easily in JAD session #2.

The group dynamics during the Define phase reflected the process people go through in learning to participate in a process of exploration of possible options. It requires people to set aside preconceived notions of what the solution should look like and that is hard to do. JAD session #1 provided a context for the team to finalize the problem definition and evaluation criteria. That freed the team from Define and allowed movement into Design.

This highlights three key lessons:

- The Define phase has similarities to a JAD session. There are ground rules. The participants need to participate, and it is important to facilitate a group dynamic that provides a complete airing of ideas and wide contribution. It is also important engender commitment among the participants. One approach to

achieving better closure during the time allocated for Define is to enforce that no one leaves on day two until the entire group can align with the problem definition and evaluation criteria.

- Otherwise, if a phase is not functionally completed, the project team and stakeholders will be going back and completing it later, and their ability to progress in future phases could be hindered until the former phase (or prerequisite elements of it) are completed.
- As a safety valve, frequent opportunities to bring the stakeholders and project team together are crucial until both groups feel they have clearly defined the needed results of that phase. Had the groups not re-synced at JAD session #1, the project would have been in trouble. If the project team feels there is troublesome ambiguity in the outcome of the Define phase or any other phase, they must ensure other sessions with all relevant parties in order to push through to clarity.

Business Action Framework – required augmentation this time

The Business Action Framework is a method for identifying and prioritizing problem areas in the context of business processes. It is used in the Define phase to select and sequence areas to address through the current and successive 30-Day Blitzes. The reasons to use the Business Action Framework are (i) that it ensures coverage of the interests of *all* of the stakeholders and (ii) that it creates unified ownership of prioritization and decisions about the system to be built.

In the Define phase, stakeholders had difficulty describing the problem context through the mechanism of the Business Action Framework (BAF). They wrestled with the process itself instead of the way the process should act. Being agile, we adapted our approach for problem definition to one in which the specific known problem areas were listed and methods were discussed for collecting data to better analyze these problems. The advantage of this shift is that it got the team away from wrestling with process and enabled forward movement.

The disadvantage is that it neglected the two main reasons for the BAF approach – representation of group-wide interests and subscription of the whole group to the prioritized results. The outcome is that the solution took a shape more focused on a technical data collection system than on improvements to business operations.

Why was the BAF a difficult process for stakeholders to use? Negotiating and renegotiating contracts and incentives with supply chain partners is where the greatest impact is achievable, but the purview of the people in the room was far more technical.

We believe this technical orientation may be endemic in the company. Rather than devise solutions to the business problems that create poor data quality and lack of visibility into supply chain operations, the business people responsible for solving the problem are empowered only to deal with the *symptoms* instead of the *causes* of the business problems. Discussing the business operations was perceived by this group as fruitless, as it was an area in which they could see little opportunity for improvement from their own actions. So the team focused on what they felt they could accomplish.

In future 30-Day Blitzes, the BAF should still be attempted, particularly in sessions with other business groups. Flexibility and responsiveness on the part of the facilitation team will be essential in order to interpret when BAF needs to be supplanted by alternate processes.

Letting go of perfection – “robust 80% solution” is difficult for some

In an organization where individual work quality is emphasized or where people employ personal standards of excellence, it can be difficult to shift perspective to the value of a “robust 80% solution” that only addresses part of a problem. We observed that some within the organization faced this struggle – to let go of some of the activities they traditionally view as part of a “complete” job.

It is crucial for the system builder and project team to perceive and manage these tendencies, as attempting to do too much in the first blitz can get in the way of achieving the timely completion of the first version of the solution system. It is understood that there are always additional system features that could be done (if prioritized) after the first version of the solution system is delivered; the main focus has to stay fixed on delivering a working system with *tangible value* in 30 days. Some approaches to addressing team members’ desire do more than needed:

- Regular reassurance that the whole team owns solution and the whole team sets the boundaries and functionality of the system in each iteration
- Coach the team with the message that any design or building for un-named, anticipated requirements jeopardizes completing the scope of *this* 30-Day Blitz; discuss individual steps and activities with this filter in place.

Roles – roles may be blended

We observed that there is no role of individual project team leader. This role was performed by a program manager who was skilled well beyond the running of one

single team. This program manager role is similar to what is called a “System Builder” in the Define-Design-Build process. The “system builder” role in this 30-Day Blitz took on some of the responsibilities normally fulfilled by the individual team leader. The system builder on this 30-Day Blitz worked hard to make the most of the skills available on the team and maximizing the contribution of each participant. Normally this work is done by the team leader and the system builder would coordinate and guide the actions of several project teams working together on a single project.

Based on this experience, we have some suggestions for the team leader role in future:

- On a multi-team project, select a qualified senior member of each team to be the team leaders for their teams.
- Plan to schedule a portion of the team leader’s time on technical activities; the team leader is a hands-on leader working with the team members and often takes on the most demanding work of the team.
- Plan to schedule some of the team leader’s time as back-up or contingency for other tasks that face completion risk or that are not anticipated and get added to the plan as the project progresses.
- On a larger, multi-team project, the system builder’s time is spent working with the team leaders to manage the activities of several project teams on the project. A well trained system builder can serve to coordinate and guide up to four or five project teams working in parallel on a larger Define-Design-Build project.

Object Model – non-software projects still need a visual, modular model

During the Build phase of a project, many parallel tasks take place simultaneously. Good project coordination helps to ensure that each required task has an assigned owner and that no single team member has more than can be done within needed timeframes. We were fortunate in this project to have a team member who delivered strong project coordination and planning that kept the plan up to date, addressed risks and issues, and effected solid resource leveling. A visual model of the technical work to be done in the build phase would have made this easier and more team-empowering.

A visual, modular map of the work to be completed enables the entire team to contribute to managing progress and minimizing critical path risks. Beyond design-value, this is part of the value of an object model. In projects where an object model does not apply to the paradigm of what is being built (as in a database solution), an alternate “model” should be derived and employed wherein the units of the solution are graphically represented for people to self-assign and communicate completion.

A partial example of this approach came during the testing/tuning stage of the project. The daily test plan showed each element to be tested (in an end-to-end context), the responsible party, and colored shading for pass/fail status at the end of each day.

Overlapping Design and Build – it might be okay

In software development projects, an object model is completed before Build begins. There is a clear delineation between Design and Build, and we emphasize that this should be adhered to. However, in this first 30-Day Blitz, the demarcation between Design and Build was knowingly staggered. Because of the built-in dependency of a data mart on a data warehouse (and this project was building both), some data mart design activities could not take place until the data warehouse design was established. Data warehouse build activities started before the end of Design, and data mart design activities continued into Build. This was a good use of the team's time.

In the context of this first 30-Day Blitz, the dependence was identified and carefully managed to ensure completion within the 30 days. Even though data mart design continued into Build, the data mart build activities were sufficiently described and pre-planned to stay within the 30-day timeframe.