EXTREME SCOPING™: An Agile Approach to Data Warehousing and Business Intelligence

by Larissa T. Moss

It is not uncommon for seasoned project managers who use a traditional methodology on a DW or BI project to feel completely out of control. The requirements appear to be a "moving target;" communication between staff members takes too long; assigning tasks in a traditional way seems to result in too much rework; and so on. To top it all off, the businesspeople are pressuring the project teams for quick deliverables (90 days or less) as they are still "fine-tuning" their requirements. As the project team scrambles to meet those expectations, data standardization is skipped, testing is cut short, documentation is not done, and quality is compromised. Sound familiar? So, how can you "have the cake and eat it too?" In other words, how can you do it right and still deliver in 90 days? You have to set aside some of the traditional project management disciplines and try a new approach.

My agile approach Extreme Scoping™ and traditional waterfall methodologies have one thing in common. They both support the notion that the earlier you catch mistakes in the development process, the cheaper and less complicated it is to fix them. Waterfall methodologies accomplish this with a rigorously followed sequence of deliverables: project charter (planning document), requirements definition (requirements document), external specifications (analysis document), internal specifications (design document), code reviews and testing, and finally user acceptance test (UAT). Each of these deliverables must be signed off by the businesspeople in order to control scope creep. But as we all know, there are always changes to the requirements, even on traditional operational system projects. Traditional waterfall methodologies discourage scope creep, especially in the later phases when the application is being coded and tested. After all, making changes while you're still designing the application on paper is bad enough, but once you have to touch the code – well, you might as well forget your project deadline now!

Unfortunately, in the DW and BI world, it's all about change. It's about ***constant*** change. Therefore, using the traditional approach, we would never get off paper – much less deliver anything in 90 days or less. We need a different approach, which means we need to change how we organize our projects and how we use a methodology.

Software release concept

The first thing to do is to break the habit of delivering an application with one project. While most organizations realize by now that they cannot build an entire DW all at once, they don't realize yet that they cannot and should not build an application all at once either. Why? Because the requirements are most likely unstable; the scope is probably too

large or the deadline is too tight; some technology components might be unproven; data integration complexity is most likely not fully understood; not to mention the quality of the source data, which probably has not been vigorously profiled because many (if not most) business rules are not documented or even known; the project team may be too small or too large or untrained; and the businesspeople are probably too busy to be full-time members on the core team. With a project like this, following a traditional methodology is the "kiss of death." You have to "get physical" as quickly as possible. That means prototyping, prototyping, and more prototyping. But how do you prototype an entire application without falling back into the waterfall sequence of requirements, analysis, design, coding, testing? You'd be right back where you started.

Here's how you do it. You break the application into multiple software releases. The scope of each software release will contain only a ***small*** portion of the application requirements, hence the name "extreme scoping." Each software release becomes a project. Each project is developed using an iterative prototyping approach. Each prototype is time-boxed (anywhere from one to three months). Most software releases should produce a production-worthy deliverable (although there are occasions where you want to further refine ["refactor"] a deliverable in the next software release before putting it into production). After several software releases, you will have a completed and fully-functioning application.

The quality of the application will probably be higher with this approach than with a traditional waterfall approach because mistakes can be found and fixed early in the development process! Except with this approach we have sliced the development process vertically across the application scope instead of horizontally across the development phases. This allows the requirements to be refined with each software release until they become stable; the scope of each software release can be small enough to fit into a tight deadline; technology components can be evaluated early on; data integration complexity can be handled in small increments; the quality of the source data can be vigorously profiled because you will incrementally find and document the business rules; the project team can be small or even in training during the early software releases with minimal impact on the entire application; and businesspeople usually become more involved in prototyping activities than they are on traditional projects where they hand over their service requests and wait for UAT. In addition, many mistakes cannot be found on design documents and would only be found once the entire application is in production and being used by the businesspeople. With the Extreme Scoping™ approach, these mistakes can now be found and corrected during the development process before the entire application is completed. And the "icing on the cake" is that these types of projects are much more fun than traditional projects because of the self-organizing project team dynamics.

Software release planning process

Creating a project plan using Extreme Scoping™ is slightly different from the traditional way. Traditionally, the project manager reviews the methodology and extracts the activities and tasks that are needed to build the entire application into a work breakdown

structure. The project manager then uses the work breakdown structure as a guide to create a detailed project plan, which is typically arranged in the sequence of the development phases of the methodology: requirements definition, analysis, design, coding, testing, and implementation. The detailed project plan is usually a 30-40 page Gantt chart, which is used to guide the day-to-day work activities, manage the change control process, and report the project status to management.

With the Extreme Scoping™ approach, the project management function is performed by the entire core team (the makeup of which I will describe below), not by a single project manager. The core team members start out by reviewing the methodology and selecting the activities and tasks that are needed to complete the entire application into a preliminary work breakdown structure. Using this work breakdown structure as a guide, the core team members create a high-level project plan (not 30-40 pages) to give them broad estimates and an understanding of the overall effort, resources, cost, schedule, risks, and assumptions for the entire application. This is necessary in order to come up with the right number of software releases, the right sequence of those releases, the dependencies among the requirements, and thus, the deliverables and scope for each release. Without this crucial step, the process of breaking an application into software releases would be completely arbitrary, with no foresight and no control over the whole development process for the entire application. It would be like throwing darts at a calendar and picking the dates hit by the darts as your software release dates.

Once the core team members are comfortable with the scope and sequence of the proposed software releases and are confident that each software release is doable within the allotted time-box (deadline), they create a detailed project plan with weekly milestones for the first software release. Starting with the deadline and working backwards, the core team members determine how far along they must be the week before the deadline in order to make the deadline, or put another way, they determine in what state the project or deliverable must be the week before the deadline. They number and describe that milestone, and then repeat the process by backing up another week and another week and so on. If they pass the project start date, the core team members must determine if the scope is too large for the deadline or if the time periods between the milestones are overestimated. All team members must agree that the milestones are achievable, and that quality will not be compromised, given the scope and resources.

After the project activities for the first software release are organized into milestones, the core team members self-organize themselves into the appropriate number of work teams. Knowing the makeup of the work teams and knowing the weekly milestones, the core team members decide on the detailed tasks and task deliverables for each milestone (referring to the work breakdown structure they created earlier). They also decide which tasks and deliverables are assigned to what person on what work team. The detailed daily task assignments and task deliverables are documented on a white board, a flip chart, a spreadsheet, or other informal media, which can be modified quickly and easily. (Detailed 30-40 page project plans created with professional project management tools are too cumbersome to maintain.) The core team members use this informal detailed project plan on a daily basis to guide the day-to-day work activities, manage the change

control process during prototyping, and monitor the progress of the project. They do not use this detailed plan to report the project status to management. Instead, they create a short one-page Gantt chart showing only the milestones. If a milestone is delayed and the core team members are confident that they can catch up the following week, the delay is not reported to management. However, if the delay has a domino effect and the next milestone is also missed, then the delay gets reported to management along with a proposed course correction.

If the first software release was completed on time and without unanticipated problems, the core team members can plan the second software release in the same manner. However, if there were problems with the first software release, such as underestimated tasks, incomplete deliverable, friction on the core team, constant adjustments to the scope, and so on, the core team members must review and adjust the high-level project plan for the entire application. They must revisit their broad estimates and their understanding of the overall effort, resources, cost, schedule, risks, and assumptions of the entire application. Then they must make the necessary adjustments to the remaining software releases. That can include changing the scope for the second software release, changing the number of software releases, reprioritizing and changing the sequence of the software releases, changing the deliverables for one or more software releases, changing the deadlines, or changing resources. Only then can the core team proceed with the detailed planning of the second software release.

Self-organizing project teams

Traditional teams depend on the project manager to coordinate and assign tasks to individual project team members and to track and report the progress of the project. The project manager also reviews the work deliverables and makes project-related decisions. The individual project team members are responsible for their own task deliverables, which they hand off at the appropriate time. For example, the requirements analyst gathers the application requirements, which are then handed off to the data modeler who creates the logical data model, which is then handed off to the database administrator who produces the physical data model and builds the database structures, which are then handed off to the developer who constructs the application. The only collaborative interactions that involve the entire team happen once a week during the status review meetings. This approach works well on very large, multi-year projects that use traditional methodologies and have 20 to 30 people on a team, but it is ineffective for DW and BI projects that must be agile and deliver in 90 to 120 days.

The following figure (Figure 1 – DW/BI Team Organization) illustrates how an agile self-organizing project team is organized.

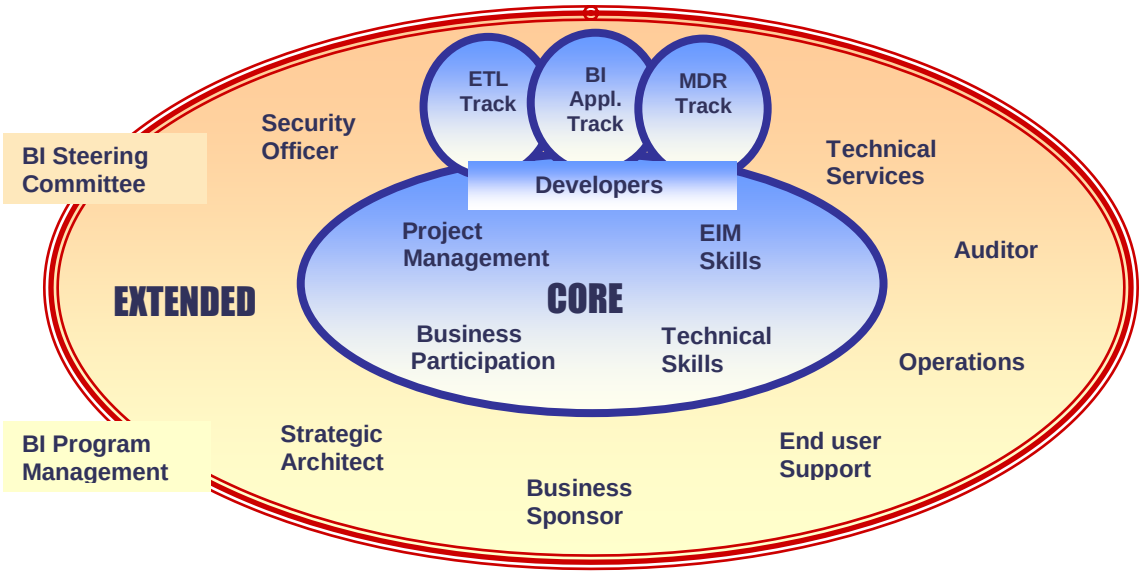

Figure 1 – DW/BI Team Organization

Core Team

The core team should be staffed – at a minimum – by a seasoned project manager; a business representative or subject matter expert who has the authority to make decisions and to set policies; an enterprise information management (EIM) person (like a data administrator) who is trained in data administration disciplines, such as logical data modeling (not database design), normalization rules, data standards, taxonomy; and at least one senior technical person (lead developer or technical architect) with strong programming and technology skills. All core team members together comprise the project management team, which replaces the single project manager function. The core team members manage all software releases of an application, thus ensuring continuity and knowledge transfer.

Core teams are self-organizing SWAT teams where all members of the team (together as a team) coordinate and assign tasks to each other. They review each other's work, collaborate on project issues, and make project-related decisions. The entire core team meets every day for a status review, and the core team members take over each other's tasks when needed (for example, when a core team member is out sick).

A core team must be very small; a team size of 3 to 5 people (per project) is optimal. Project core teams should never exceed 7 people.

Development Track Team

The most common development tracks are ETL development, BI application development, and metadata repository (MDR) installation or development. If metadata is not a deliverable, then at least two development track teams work in parallel, namely ETL and BI application development. These development track teams are extensions of

the core team. However, the development track team members do not participate in daily project management activities.

The development track teams are led by the lead developer or technical architect from the core team. You may also choose to have a lead developer or technical architect per development track, if that better supports your organization and if you have the resources. Development track team members should be developers (and potentially a senior systems analyst) that have the skills needed for their particular track. They only participate in track-specific tasks for the duration of their track activities.

A development track team is even smaller than a core team; a team size of 2 to 3 people (per track and per project) is optimal. Development track teams should never exceed 5 people.

Extended Team

The extended team members play traditional roles and participate on the DW and BI projects on an as-needed and as-scheduled basis, not on a full-time or daily basis. Members of the extended team may be heavily involved only at certain times during the project, or they may be called intermittently for their expertise and contributions. Extended team members include technicians and businesspeople who contribute in much the same way they do on traditional projects. Extended teams include support roles such as technical support, operations, the security officer, IT auditor, the business sponsor, and other stakeholders.

The core team, the development track teams, and the extended team make up the complete DW/BI project team.

BI Steering Committee

The BI steering committee is an advisory body of business executives and senior business managers, who understand BI and DW, who support enterprise-wide activities, and who provide collective sponsorship. They meet on a regular basis to discuss, plan, staff, and fund business initiatives, such as DW, BI, MDM, CRM, CDI, and so on. They communicate necessary data integration principles to the organization. They stand behind a DW/BI strategy that supports the business drivers. They fund an enterprise information management group to perform enterprise-wide data governance activities. They also free up businesspeople from their operational responsibilities so that they can participate as full-time members on the project core teams.

BI Program Manager

The BI program management office is led by a BI program manager (director) who works directly with – if not for – the BI steering committee. The BI program manager performs periodic readiness assessments to identify new information needs and to ascertain the satisfaction among businesspeople with the DW and their BI applications. With the

backing of the BI steering committee, the BI program manager creates a BI strategy and enforces the common technical and non-technical infrastructure components in the BI environment. Working with the BI steering committee, the BI program manager prioritizes BI and DW projects, determines the project interdependencies, and coordinates the project resources and activities around these interdependencies.

Team Interaction and Communication

The dynamics of the core team and the development track teams are not the same as that of the extended team or a traditional project team. The core team is like the SWAT team at NASA in Houston, Texas. Although they have their assigned cubicles and offices, they do not spend most of their time in them. Instead they collaborate, brainstorm, solve problems, and make decisions together in a "war room" dedicated to their project. For example, the core team meets every day to review status and deliverables and to discuss roadblocks and solutions. Sometimes these meetings last ten minutes, at other times working sessions could be scheduled for half a day. During these sessions, individual assignments are distributed to the appropriate team members who will work on them right after the meeting and report back the following day. If major roadblocks are encountered, they are addressed immediately and alternative solutions and contingency plans are prepared. The project manager and the business representative will take the alternative solutions and contingency plans to the sponsor for negotiation and/or approval.

The development track teams function similarly to the core team, only on a smaller scale. They also meet every day to review the status and deliverables within their own track. They collaborate, brainstorm, solve problems, and make decisions together on their track-specific issues. In fact, they function similar to extreme programming (XP) teams sharing the prototyping activities of analysis, design, coding, and testing. Communication with the core team is automatically established through the lead developer or technical architect, who is managing the development track teams.

The extended team meets with the core team once a week, usually on the same day and time of the week. The core team members set up the meeting schedule at the beginning of the project, and the sponsor sends out the notification to the extended team members. The meetings are scheduled for one hour, but could be as short as ten minutes if no major issues are to be discussed. The purpose of the weekly meetings is to keep all team members current with activities, status, and issues of the project.

Conclusion

In summary, the popular concept of "iterative development" applies not only to developing the DW in small increments, but it should be practiced at the application level as well. While traditionally the definition of a project was to deliver a completed, fully-functioning application, in an agile approach a project equates to a software release, and it takes ***multiple*** software releases, i.e. multiple projects, to deliver a completed, fully-functioning DW/BI application. In addition, you have to convert your traditional project

teams into agile, self-organizing SWAT teams. The key is to keep your project teams small. Transfer the project management activities to the core team members (collectively). Be sure you have a business representative on the core team. Your development track teams should be pair-programming teams of 2-3 developers. And finally, include the extended team members on your project on an as-needed basis.

If you want to learn more about Extreme Scoping™, you'll have to wait for my next book. I am hoping to publish it in the fall of 2010. In the meantime, come and attend the project management and methodology seminars that I teach at various conferences in the US and Europe.

Author's BIO:

Larissa Moss is founder and president of Method Focus Inc. She has 30 years of IT experience, with a focus on data warehousing for the past 22 years. She frequently speaks at conferences worldwide on the topics of data warehousing, business intelligence, project management, development methodologies, and other information asset management topics, such as enterprise architecture, data integration, and information quality. She co-authored the books *Data Warehouse Project Management*, *Impossible Data Warehouse Situations*, *Business Intelligence Roadmap*, and *Data Strategy*. **She is currently working on her next book *Extreme Scoping™: An Agile Approach to Data Warehousing and Business Intelligence*.**

Her articles are frequently published in*Teradata Magazine*, *TDWI Journal of Data Warehousing, TDWI Flash Point, Cutter IT Journal,* and *EIMInsight Magazine* at www.eiminstitute.org. Her white papers include *Organizational and Cultural Barriers to Business Intelligence*, *Developing BI Decision-Support Applications: Not Business As Usual, Data Quality is Not Optional*, and *The Importance of Data Modeling as a Foundation for Business Insight*.

Her present and past associations include Third Party Influencers of Teradata, the IBM Gold Group, the Cutter Consortium, DAMA Los Angeles Chapter, the Relational Institute, and Codd & Date Consulting Group. She was a part-time faculty member at the Extended University of California Polytechnic University Pomona, and has been lecturing for TDWI, DCI, PESG, MIS Training Institute, and the Cutter Consortium. She can be reached at methodfocus@earthlink.net.

