Comparative Analysis of Business Information System Development Methodologies

Table of Contents

1.0 Objective ........................................................................................................... 1

2.0 Waterfall System Development Life Cycle ....................................................... 1

3.0 Agile System Development Life Cycle ............................................................. 3

4.0 Case Study: Agile-based Iterative System Development Approach ................. 6

5.0 Data-Centric System Development Life Cycle with Prototyping And Code Generation Approach ................................................................. 10

6.0 Summary Advantages from Melding the Waterfall and Agile Methodologies into the Data Centric Business Information System Development Life Cycle with the Metabase System, Prototyping And Code Generation ........................................ 12
   Advantages Inherited from the Waterfall Approach ......................................... 12
   Recast Disadvantages as Advantages Inherited from the Waterfall Approach ..... 13
   Advantages Inherited from the Agile Approach .............................................. 13
   Recast Disadvantages Inherited from Advantages from the Agile Approach .... 14

7.0 References ......................................................................................................... 14
1.0 Objective

The objective of this short paper is to provide a brief overview and comparison of different business information system development approaches, and to identify through a case study the consequences when an approach fails. The topics addressed are:

- Waterfall System Development Life Cycle
- Agile System Development Life Cycle
- Iterative System Development Approach via Agile
- Data-Centric System Development Life Cycle Coupled with Code Generation

For the purposes of this paper, the phase, implied steps and likely work products are the same as those specified in the Whitemarsh Short Paper, “Business Information Systems Life Cycle Costs.” It is located at www.wiscorp.com/sp/sp026.pdf. There are a number of additional relevant short papers listed in Section 6, References.

2.0 Waterfall System Development Life Cycle

Figure 1 illustrates the general flow of the waterfall approach. From this figure it is clear that there are no feedback mechanisms. At each stage, 100% of the work products appropriate for that stage for the entire business information system are created, reviewed, revised, and finalized. At that point, that specific phase of the project is closed and the next phase starts. Although not shown in Figure 1, the first phase would include a data requirements set of activities that would create a data model commonly be called a Logical Data Model. That is, one that contains a complete set of database tables specifications including columns and relationships. Likely not done would be the Physical Data Model because its design would not have been changed to fit the specific needs of the DBMS nor the performance needs of the business information system.

The key benefits of the waterfall methodology, as adapted from materials from the Base36 organization (http://www.base36.com) include:

- Precise specification development across a well defined set of methodology work products.
- The stake holders of the business are able to know the steps of the methodology and the appropriate time-line and resources necessary for completing these work products.
- As a phase is completed, it can be turned over to a different development group with the assurance that 100% of the work product artifacts are complete, reviewed, and finalized.
Figure 1. Graphic depicting the architecture of the Waterfall Methodology.

The key disadvantages of the waterfall methodology include:

- Once a step and/or a phase containing a series of steps have been completed and turned over to a follow-on set of developers, they do not go back to a previous step or phase and make changes. Simply put, the scope of the follow-on set of developers is restricted to the documents created during the previous phase.

- The Waterfall methodology relies almost exclusively on the creation of a set of requirements acquired from end-users and state-holders. Regrettably, if the requirements are incomplete, or faulty, the project’s success possibility is compromised.

- Whenever a requirement is discovered but missing or is in error, the necessary fix is set aside for a future maintenance cycle after the completion of this specific round of business information system development.
Comparative Analysis of Business Information System Development Methodologies

- The whole business information system is tested only at the end. If an error or bug is discovered that was created in an initial component of the business information system, the entire testing process would likely terminate and not recommence until the bug is fixed. This process could greatly extend the whole testing schedule.

- The waterfall methodology cannot accommodate undiscovered and/or evolved needs. If such a need is discovered, there are only two alternatives. First, stop the entire project and return to the first phase, or put off the undiscovered requirement until a business information system maintenance cycle.

3.0 Agile System Development Life Cycle

The Agile approach is almost the inverse of the waterfall methodology. Agile came about as a “solution” to the disadvantages of the waterfall methodology. Instead of a sequential design process, the Agile methodology follows an individual component approach. Figure 2 illustrates the set of processes typically included in an agile life cycle. While there can be many names for these processes they are essentially the same as those in the Waterfall methodology. The only change is that Requirements Gathering is factored out from the phases set out in Figure 1.

The key difference in Figure 2 from Figure 1 are that there is an implied cycle of the processes from Requirements Gathering through to Feedback and onward to the next cycle of Requirements Gathering. This change enables requirements to be handled as they are discovered.

At the center of Figure 2 is an identified functional component from the overall business information system. This identified functional component is uncovered through an overall process of requirements gathering for the entire business information system. Once an overall level of requirements are discovered, they are subdivided into homogeneous collections and are set out into a precedence-based sequence that generally ensures that the business information system is created in an orderly manner. The process of implementing these identified function begin through the encapsulated agile processes shown in Figure 2. This overall business information system agile sequence strategy is illustrated in Figure 3.

Fundamentally, for each identified functional component, developers start off with the determined overall project design, and when starting to focus on an identified component, the work set out in Figure 2 is accomplished in weekly or monthly process collections called sprints. At the end of each sprint, project priorities are evaluated and tests are run. These sprints allow for bugs to be discovered, and customer feedback to be incorporated into the design before the next sprint is run.

The agile methodology is collaborative in nature and is built around small teams of staff with different but needed skill sets. Team engineering was championed in Fredericks Brook’s 1975 book, The Mythical Man Month. Chapter 3 of this book, The Surgical Team, sets out the names and descriptions of all the different participants in an IT team.
The agile process, with its lack of initial design and steps, is often criticized for its collaborative nature that focuses on principles rather than process. There is merit to this approach however, because otherwise team members become slaves to the precise tasks as opposed to common sense.

There are a number of advantages to the agile methodology including:

- Changes are allowed to be made after the initial planning including complete redesigns and rewrites of computer software to reflect the unfolding needs of the client and also the refinements of the client’s understanding of the problem to be solved.

- As a consequence of enabling in-process changes, the overall specification and implementation of a business information system is likely to be more correct on its first production version.
Because the overall business information system project is broken into smaller functional components that are implemented in groups, as each is completed, there can be a reassessment of the build sequence and various project priorities are evaluated. This allows clients to add their feedback so that they ultimately get the product they desire.

Instead of waiting until the very end to accomplish end-to-end testing, as the software modules are created within each of the functional components, bugs are discovered, addressed, and are resolved within the agile cycle for that functional component.

Functional components are engineered and configured to be useful immediately upon their completion. This enables the business information system to become operational incrementally rather than only at the very end.

As with the waterfall methodology, there are also disadvantages to the agile approach. These include:
Because agile is fundamentally not top-down but bottom-up, the agile process often appears to be the “herding software programming cats.” With a less successful project manager, the project can become a series of code sprints that may become out of control because the different program modules have overstepped their natural boundaries, encroached on the domains of other functional components, or even developed computer software that is in significant conflict with other developed software. When this happens, the project often comes in late, over budget, or not at all.

Because there is no definitive and precise specification of the business information system from the very beginning as there is with the waterfall approach, the final business information system product can be significantly different than what was initially intended.

Because each agilely developed functional component is encapsulated, the processes involved with the creation of a functional component contained data model takes on the form of an application/module-based data model that can contain conflicting table collections, columns, data types, value domains, and relationships.

4.0 Case Study: Agile-based Iterative System Development Approach

During the 2009 - 2010 time frame, an actual implementation of the agile approach was undertaken on a large scale business information system project. The project, which ultimately failed had a cost of about $110 million. The project was based on an overall set of requirements contained in a document in excess of 100 pages.

After a detailed review of the requirements document, the development team set out a collection of functional components and sequenced them for development. Each functional component was known as a package. Within each package there were identified a collection of products. Once all the products of a package were completed, the overall package was set into production. After all products of a package are complete, the overall package is moved into production. The Package Specification Life Cycle figure below illustrates this sequential process.

Figure 4. Package Specification Life Cycle
the packages were complete, the overall business information system was deemed implemented.

During this initial requirements review and functional component identification effort, an overall business information systems data model was created and was issued to development.

Figure 4 depicts the five distinct components of each package. For each there was a general work breakdown structure. In this figure, the initial specification included the creation of a series of use cases. Upon review, revision, and acceptance, an activity diagram was created for each use case. From the activity diagram a set of wire-frames (computer screens) were created. Finally there was a logical data model created to reflect the requirements of the wire-frames (screens) and the fields in the processes. In addition to the logical data model, a data dictionary was created that contained specifications of all data elements for the screens.

Figure 5 illustrates the creation of an overall product’s specification. This figure shows that there is interaction among the packages. Each package was required to be properly integrated, and as required, updated with the other packages.

![Diagram of package integration](image)

**Figure 5.** Product Specification Life Cycle.
Once a package specification was complete, its implementation begins. Figure 6 illustrates the overall process model for implementation. This starts with the collections of Use Cases, Activity Diagrams, Wire Frames, and Logical Data Model from each package.

There is then a design process that produces the IT-based work package listed in the Design Stage box of Figure 6. Once these are transformed to their appropriate computer-software form, the package is delivered to the Testing stage. During testing, any failures are cycled back to the Design Stage for correction and if necessary to the package’s specification as contained in Figures 4 and 5.

Ultimately, as all the packages are developed, as illustrated in Figure 7, the business information system emerges as the various packages are placed into a operational production
status. As stated at the outset of this section, the overall business information system failed. Not just partially, but completely. The reasons for failure are essentially three.

First, the employed agile business information system development methodology never created an end-to-end demonstration of the business information system’s functionality prior to the creation of production class software. Because end-to-end prototyping was not done, problems surfaced in the later packages that ultimately required changes to the data models of earlier packages that in turn required significant production-level computer software code.
Comparative Analysis of Business Information System Development Methodologies

changes. Such changes required modification of package specifications along with significant changes to the implementation of these packages.

Second, there was never an end-to-end data model that became the basis for prototype generation and which as able to be continuously updated as the packages were created and demonstrated through prototyping and then revised.

Third, the vast majority of the software program code was hand-coded as opposed to generated. That alone eliminated the ability to cost-effectively create the prototypes essential for the validation and evolution of the overall architecture and design of the business information system.

5.0 Data-Centric System Development Life Cycle with Prototyping And Code Generation Approach

The data centric system development life cycle founded on code generation represents a merger the positive characteristics of both the waterfall and agile development methodologies and also an avoidance of the disadvantages of both these approaches. This approach is described in Short Paper 26, and so is not be duplicated here.

The remainder of this section addresses the combined set of advantages and disadvantages of the Waterfall and Agile development approaches. Note that the disadvantages are recast to become advantages.

Figure 8 illustrates the overall life cycle for this approach not including the maintenance cycles.

![Diagram](image)

**Figure 8.** Data Centric System Development Methodology including Prototyping.
Figure 9 shows the integration between the various models contained in a Data Architecture Reference Model and the collection of work products that support the specification, implementation and maintenance of business information systems. This figure shows how all the work products across the specification, implementation, and maintenance of the business information system are integrated with the collection of data models.

Figure 9. Data Model Center to the Business Information System Universe
Comparative Analysis of Business Information System Development Methodologies

Figure 10 illustrates the complete data-centric business information system approach. The left side of the figure shows that 33% of the overall effort is allocated to the creation of the Version 4 design.

![Data Centric Business Information System Development Methodology including Maintenance Cycles.](image)

Figure 10. Data Centric Business Information System Development Methodology including Maintenance Cycles..

A Version 5 design results from the feedback on changes that occur after prototype #4. 66% represents the amount of effort that must be expended to accomplish implementation of the first production version. Subsequent to that, there are three cycles of maintenance instead of the usual five.

6.0 Summary Advantages from Melding the Waterfall and Agile Methodologies into the Data Centric Business Information System Development Life Cycle with the Metabase System, Prototyping And Code Generation

Advantages Inherited from the Waterfall Approach

- Precise specifications are developed across a well defined set of methodology work products.
Comparative Analysis of Business Information System Development Methodologies

- Stake holders of the business are able to know the steps of the methodology and the appropriate time-line and resources necessary for completing these work products.

- Prototyped specifications can be turned over to a different development group with the assurance that 100% of the prototyped work product artifacts are complete, reviewed, and finalized.

Recast Disadvantages as Advantages Inherited from the Waterfall Approach

- Once a step and/or a phase is completed and turned over to a follow-on set of developers, they can go back and quickly generate a revised prototype that can be reviewed and revised.

- Requirements acquired from end users and state-holders can be employed as discovered and folded into prototyping activities.

- Whenever a requirement is discovered that is missing or is in error, prototyping and code generation enables quick maintenance and the regeneration of the next set of prototypes.

- Business information system functionality can be tested as it is prototyped. If an error or bug is discovered, it can be quickly fixed in the work products and a new prototype generated.

- Undiscovered and/or evolved needs can be accommodated either by changing the prototypes or by putting off the undiscovered requirement until a maintenance cycle.

Advantages Inherited from the Agile Approach

- Changes are allowed to be made after the initial planning including complete redesigns and generation of prototyped computer software that reflects the unfolding needs of the client and also the refinements of the client’s understanding of the problem to be solved.

- As a consequence of prototyping, the overall specification and implementation of a business information system is likely to be more correct on its first production version.

- Based on enterprise mission specification and mission-based data modeling, the overall business information system effort can be quickly broken into smaller functional components that are implemented in groups. As each is completed, reassessment of the
build sequence and various project priorities can be evaluated. This allows clients to add their feedback so that they ultimately get the product they desire.

- As the software prototypes are completed, bugs are readily discovered, addressed, and are resolved with the next generation of that functional component.

- Functional components are engineered and configured to be useful immediately upon their completion. This enables the business information system to become operational incrementally rather than only at the very end.

Recast Disadvantages Inherited from Advantages from the Agile Approach

- Top-down prototyping enables end-to-end management of the business information system effort. Code generation eliminates the need for code sprints that can become out of control because there exists an over-arching end-to-end set of work products that are integrated, interoperable and non-redundant that are stored in the Metabase System database.

- There is a definitive and precise specification of the business information system from the very beginning because of the Metabase System, prototyping, and code generation. Thus, the final business information system product maps to what was ultimately intended.

- While each developed functional component can be encapsulated, the processes involved with the creation of a functional component contained data model are stored and enable interrelationships of all functional modules across the entire business information system.

7.0 References

The following references to Whitemarsh materials provide a more detailed exposition practical application of the significant content of this paper.

The following documents are available free (except for the Mythical Man Month book) from the Whitemarsh website:

<table>
<thead>
<tr>
<th>Paper</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise Architectures</td>
<td><a href="http://www.wiscorp.com/sp/sp08.pdf">www.wiscorp.com/sp/sp08.pdf</a></td>
</tr>
<tr>
<td>Paper</td>
<td>URL</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Reverse and Forward Engineering Guide</td>
<td></td>
</tr>
<tr>
<td>Metabase Module User Guides</td>
<td></td>
</tr>
<tr>
<td>Metabase Overview</td>
<td><a href="http://www.wiscorp.com/metabase.zip">http://www.wiscorp.com/metabase.zip</a></td>
</tr>
<tr>
<td>Modeling Data and Designing Databases</td>
<td><a href="http://www.wiscorp.com/sp/sp06.pdf">www.wiscorp.com/sp/sp06.pdf</a></td>
</tr>
<tr>
<td>Database Objects Introduction</td>
<td><a href="http://www.wiscorp.com/sp/sp22.pdf">www.wiscorp.com/sp/sp22.pdf</a></td>
</tr>
<tr>
<td>Mythical Man Month (Frederick Brooks)</td>
<td><a href="http://www.amazon.com/Mythical-Man-Month-Software-Engineering-Anniversary/dp/0201835959/ref=sr_1_1?ie=UTF8&amp;qid=1450296416&amp;sr=1-1&amp;keywords=the+mythical+man+month">http://www.amazon.com/Mythical-Man-Month-Software-Engineering-Anniversary/dp/0201835959/ref=sr_1_1?ie=UTF8&amp;qid=1450296416&amp;sr=1-1&amp;keywords=the+mythical+man+month</a></td>
</tr>
</tbody>
</table>