White Paper

Software Development Process Super-charged by Xsemble

Ashish Belagali,
10Xofy, Pune, India

January 2021

Abstract

IT projects are regarded as being inherently complex to manage. The challenges in managing them lead to schedule overruns, cost overruns and project failures. However, they can be addressed by using Xsemble technology, as it deals with exactly those parts of the software development process which cause these challenges. The conventional tools and techniques which address other areas well can be combined with what Xsemble offers, to get an end-to-end process which can be managed and executed smoothly. We present this enhanced process in this article. We also show 10 ways in which the new process can add value to the project management.

Table of Contents

1. Problem: Difficulty in Managing IT Projects

"IT projects are notoriously difficult to manage", states an article and gives statistics around it. Practicing Project Managers know this from experience. The article further confirms another observation that larger projects are more likely to fail than the smaller ones. The important questions are: what makes them so difficult and what can one do to make it easier?

Usually one would turn to the discipline of project management to see where it can be fixed. However, it turns out that vital information that is critical to understand how the project is going is hidden in the code and not readily available for inspection. Take for instance the question, "What is the % progress of development?" This number is most often a guesswork by the developer, in the absence of a good quantitative measure. As with any guess, some times they may be close, some times not. More importantly, in case the developer overlooked some important aspect while giving this number, then there isn't a good way of knowing that. This can result in costly schedule overruns, and it is no one's fault.

When the project management is done on the basis of error-prone guesswork, how can it deliver correct results consistently?

It is interesting to note that all areas of software development are not equally difficult to manage. Borrowing the SDLC (Software Development Life Cycle) terminology, the main problem areas are the Low Level Design (LLD) and the Development. For other areas, reliable information can be extracted from the data that is readily available, as the following examples show:

- The function points can help one make a quantitative statement about the size of the requirements.
- The Requirement Traceability Matrix (RTM) can be used to make quantitative statement about the adequacy of test cases.
- Defect density or can be used to quantitatively assess the quality of the individual feature implementations.

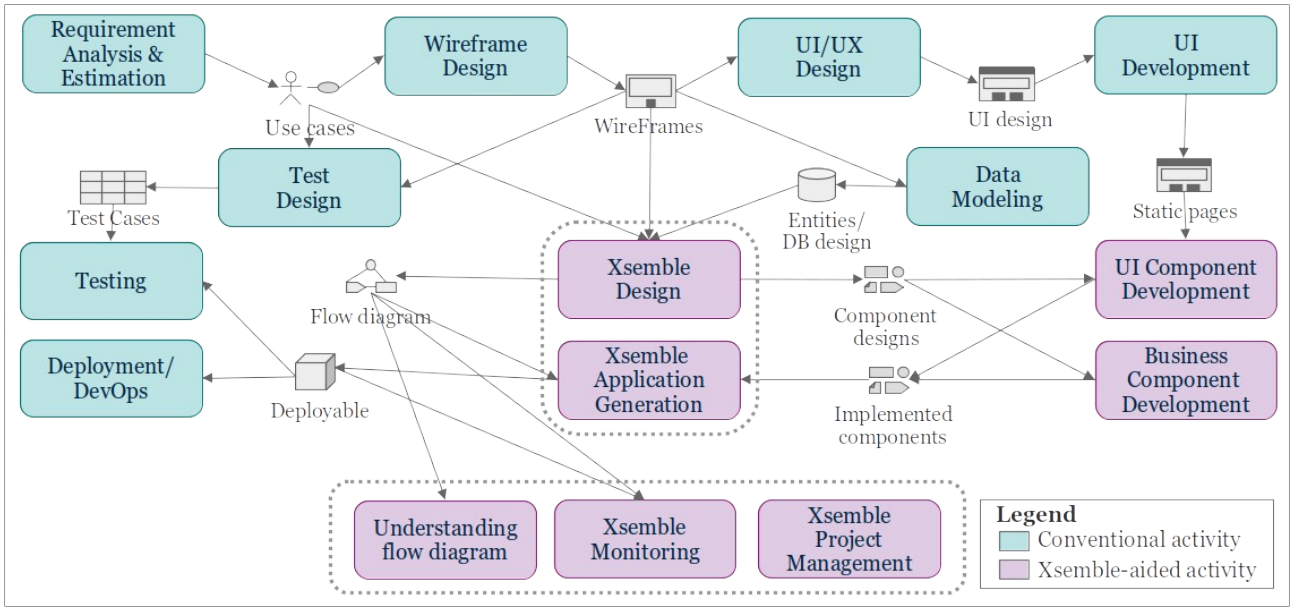
Some may point to LOC (lines of code) as a metric for the source code, but LOC is known to be a poor indicator for number of reasons, and the information extracted from it can be misleading (reference). On the other hand, using function points instead of LOC for sizing means that you are measuring the size of the requirements and not that of the implementation.

2. How Xsemble Solves the Problem

By making the design widely accessible in the form of a visual diagram, Xsemble nips this problem in the bud. It means that any stakeholders, without having to get into the source code, can now inspect the design and raise any concerns. Because this visual is guaranteed to represent a true picture, this exercise is directly relevant to the actual working of the application. It offers the one a chance to apply necessary checks and balances at the design level. The programming is pushed into the confines of tiny components. Being small, they are easier to manage and control. New quantitative measures such as number of components, their sizes and the cyclomatic complexity of the flow diagram become available. In short, we get a handle on managing the part that has been the blocker in effectively managing the software development process.

3. The New Software Development Process

The Process Flow Diagram (PFD) below captures the enhanced software development process aided by Xsemble. Each block is an activity. The background of the activities is color-coded to differentiate conventional activities (light blue background) from the new activities (light purple background). The inputs and outputs of each activity are clearly captured (with the exception of the the "Xsemble Project Management" activity where all the artifacts in the diagram are the inputs and corresponding arrows would be too numerous to show in the diagram).



In the diagram, some Xsemble activities are grouped by drawing dotted lines around them to indicate the broad skill groups of the practitioners who perform those activities.

Given below is a brief description of each activity.

3.1. Conventional Activities

1. **Requirement Analysis & Estimation**: This activity involves capturing the project requirements. That may be done in terms of use cases, user stories or sophisticated SRS (Software Requirement Specification) documents. Project estimation is carried out to evolve schedule and cost expectations. Some times, technical POCs (Proof Of Concept) may be created to assess the feasibility and to help in better estimation.
2. **Wireframe Design**: If requirements is the question, design is the answer. Wireframes are created to answer what kind of UI (User Interface) layout would satisfy the requirements. Some modern tools help one create click-through wireframes which deliver a near complete user experience to understand the behavior upfront.
3. **UI/UX Design**: This involves designing the User Interface (UI) in detail and also making sure that the User Experience (UX) will be smooth. While UI design pertains mainly to the visual aspects (color theme, fonts, spacing etc), the UX design pertains to what kind of experience the user would have with the application.
4. **UI Development**: This is a technical activity. In this, the user interface is developed in a form that it is compatible with the target device. For instance, if you are creating a web

application, then the static web pages are created such that they are compatible with the browser.

5. **Data Modeling**: In this activity a data model is evolved. It involves identifying the various entities and the inter-relationships among them. The output of this phase is typically an ERD (Entity Relationship Diagram), a Class Diagram or an Object Diagram.
6. **Test Design**: The various test cases are identified and documented in this activity. Planning would be involved to identify how much testing needs to be done across the different features of the software.
7. **Testing**: This activity is carried out to test how the given deployable performs against the test cases, and to log any defects. The testing can be manual or automated. In case of the latter, the test cases also need to be developed at the end of test design process.
8. **Deployment/DevOps**: In some cases, there may be a series of steps before a deployable gets to the production environment, and those are covered under DevOps. The focus is to do freuquent deliveries (could be even multiple times a day) with high quality.

3.2. Xsemble-Based New Activities

1. **Xsemble Design**: The goal of this activity is to create the Xsemble flow diagram(s) that represent how the software works. The flow diagram is a directed graph. Many nodes in this graph are instances of Xsemble components. These components therefore need to be defined before creating the nodes. (Some components may be pre-defined and can be used as is.) Thus, both the component designs and the flow diagram are outputs of this activity. This activity does not involve programming.
2. **UI Component Development**: This is the activity of adding implementation to the UI components that were defined in the Design activity. This implementation is typically the source code that corresponds to the dynamic behavior of the component. The developer embeds the static UI developed in the “UI Development” activity within the code templates generated by Xsemble and then makes the code changes.
3. **Business Component Development**: Better known as “Serverside Component Development” in context of web applications, this activity involves adding the business logic code to the non-UI components. The developer adds this code to the component templates generated by Xsemble.
4. **Xsemble Application Generation**: In this activity, Xsemble generates the glue code that orchestrates the working of the components as dictated by the flow diagram. The output is the complete source code of the final application which can be compiled into a deployable. (Xsemble can invoke this compilation for you to deliver the deployable directly).
5. **Understanding Flow Diagram**: The Xsemble flow diagram is the accurate and explicit representation of the actual functioning of the software. Many stakeholders will gain useful insights that are helpful in their work by going through it. For example, the functional experts may review it to ensure that it conforms to the domain requirements, and the corner cases are handled.

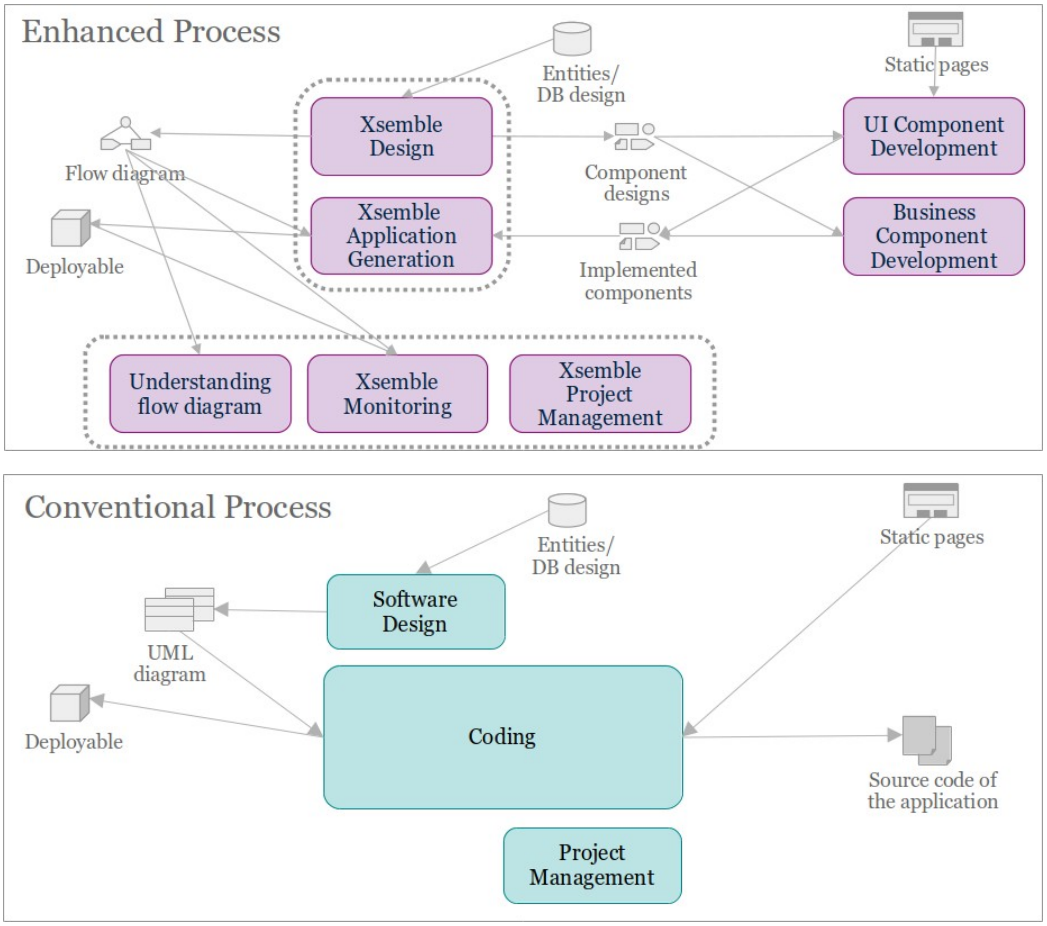
6. **Xsemble Monitoring**: Xsemble offers a live monitoring tool (Monitor's Workbench) which enhances the understanding of actual working of Xsemble through visual monitoring of a deployed application. This activity is very similar to the way programmers carry out debugging; except that here it is visual and needs no programming knowledge. That increases this process's application immensely. For instance, support engineers may use monitoring to gain insights about a customer issue, which may ease out in high quality error reporting if not the instant resolution of the issue.
7. **Xsemble Project Management**: This is an extension of the usual Software Project Management. There is richer data available, and it can be used smartly for managing the work. A few suggestions of how it may be used are presented below in the section Enriching Project Management.

3.3. Waterfall or Agile?

The enhanced process applies equally well to waterfall approach of development or the iterative approach. It is orthogonal to it. The iterative approach is the core of the agile methodologies, and consequently the enhanced process is compatible with those methodologies.

3.4. Benefits over the Conventional Process

Let us focus on the changes in the process. We first show the part of the enhanced process and then show the conventional process part that it replaces.



The number of activities and those of the artifacts are clearly more in the enhanced process, giving us a hint to the increased granularity. However, on a qualitative level there are further differences, as captured in the following table.

Conventional Process	Enhanced Process
The software design is fed to the Coding activity one time. The Coding activity can override the design decisions, and they are not reflected back into the design. Therefore, the design artifacts are not reliable.	Xsemble application generation activity is a trivial activity which includes a few clicks. It cannot change the design. The glue code is generated as per the design. Therefore, the design artifacts reflect the true picture.
Project management has less inputs. Specifically, there are no reliable inputs assess the quality and quantity of the most important activity, that is, Coding.	Apart from a reliable design diagram, the number of components, their sizes and their progress status can be great inputs to assess and control the activities.
Stakeholders have little visibility into the progress. It is limited to what the project manager feeds, which itself is based on erroneous or incomplete data.	The stakeholders can have a good visibility into the functioning and the project progress, through understanding the flow diagram and further monitoring a live application.
The modularity at the code level may or may not be maintained well over time. It is opaque to non-programmers; and difficult to assess even to programmers when the codebase grows. Code can deteriorate in quality quickly and become non-maintainable.	The code always stays modular, and a good amount of information about the components is explicitly available without getting into the code. The component level code is small, hence easy to understand and modify. Therefore the maintainability is very high.

Basically, the large and ever-growing source code gets replaced with tiny component implementations along with other artefacts which make it much easier to manage. The large coding activity which contains most of the rework gets replaced by smaller and more crisp activities. This means that the enhanced process hits exactly at those parts where conventional process created lack of visibility, rework and uncertainty. The enhanced process is therefore much easier to carry out and much easier to manage. We shall see some specific cases about the latter in the next section.

4. Enriching Project Management

Xsemble makes available more data and information. It can be used in planning, tracking, execution, risk assessment, reporting and in general all management functions. In this section, we cover 10 ways in which existing wisdom may be extended & reused for better management.

4.1. WBS in Terms of Components

When Xsemble is used, the development phase in SDLC becomes the development of components (UI and business components). As a result, the Work Breakdown Structure (WBS) is now formed out of components.

Conventional tools and methodologies used for project planning and tracking (such as a Gantt chart, Burndown chart or Kanban chart) are still applicable as before. Xsemble progress report feature can be helpful in reporting the accurate progress across all components. Xsemble also gives a ready-made Kanban chart view for convenience.

4.2. New Metrics

In scrum, the velocity of development team is determined in terms of how many function points it covers in a sprint. On similar lines, one may be able to measure the velocity of the development team in terms of the component points (the T shirt size of the components) implemented, and that of the designing team in terms of the component points added.

Further, it is now possible to track rework quantitatively in terms of the component points corresponding to the components which were previously deemed as implemented but had to undergo a change.

4.3. Team Management

The major help a manager receives is in the area of team management. This is an area where projects usually suffer highly, so any help in this area is quite important.

Managing the development resources becomes much simpler. This is a result of change of focus of the development activity from the complete application to a single component at a time. It reduces the focus by orders of magnitude from a few thousands of lines of code to a very small number of lines of code. Creating a new component or changing an existing component are very easy tasks and may be done with relative ease.

This ease further means that a new programmer will be able to become productive quickly. Replacing an existing programmer on the team would be a lot simpler now, causing low downtime if any. The manager does not need to worry about transfer of the tacit knowledge from the old programmer to the new one.

Extending this thought, one may also be able to utilize resources external to their company or the resources on bench for development if necessary. This ease of resource augmentation is a boon to a project that is falling behind schedule; and you may also be able to utilize expertise outside your company for specific components.

4.4. Efficiency Boost Through Parallelism

The iterative approach to development is defined as the one that produces incremental releases (read deployables) at the end of every iteration.

In some books, it is explained as a complete SDLC process boxed within the time frame of each iteration. However, in practice, different activities corresponding to the functionality of a single iteration are deliberately performed out of phase. Specifically, while the development of iteration N is in progress, the testing team may be testing the incremental release delivered in iteration N-1, and the requirement planning may be happening for iteration N+1. That induces parallelism in these activities, in the sense that the development team does not need to wait on the testing team to

complete their testing in the same iteration before releasing the functionality. This parallelism results in a better resource utilization and increases the efficiency of the overall process.

Now, with further breakup of the activities (as per the PFD), this parallelism may be stretched further, limited only by the readiness of artifacts which are input to it. Thus, instead of sitting idle, a UI component developer will be able to start working on a component as long as its dependencies – the component definition and static page-- are available, even if the component is not marked for the current iteration. This can be said about other activities too. Imagine how much of an efficiency boost the increased parallelism can deliver.

4.5. Change Management

Conventionally, change management is associated with a risk that the code change has undesired consequences. In other words, when you change something, the functionality could break in some other place. This happens more often with larger projects.

In the Xsemble-aided process, the change management becomes very crisp. Using the flow diagram(s), you identify exactly which components would get affected and make changes to them.

Xsemble lets you check if a given component is used in other places within the flow diagram. This means that you will be able to find out with precision which all places a change may impact, and analyze the impact upfront, even before making the change.

4.6. Project Communication

The communication to different stakeholders can become more objective – with better quantitative measures and the flow diagram being available. The stakeholders are in a better position to understand and contribute back to the project because of that.

4.7. Traceability Matrices Making Use of Components

The conventionally used Requirement Traceability Matrix (RTM) is a graph of test cases against the requirements. It becomes a good idea to add another axis of components to this arrangement. This gives rise to the following new traceability matrices:

1. **Requirements against Components**: This traceability matrix can give a great handle to understand which components contribute to implementing a given requirement, and hence assess the adequateness of the implementation. It can also be used in change management to identifying which components could change in response to a change in a given requirement.
2. **Components against Test Cases**: This traceability matrix can give you how well a component is covered by the test cases. If a component experiences more failures, then the test case coverage around that needs to be made stronger.

4.8. Extending Defect Analysis at Component Level

The parameters such as defect density, defect age may be tracked against components. The historical data on how many defects need fixes to the components can be used to pinpoint unstable

components, so that they may be re-engineered to improve overall stability of the system dramatically.

4.9. Understanding and Managing Complexity

The complexity is managed at two levels:

- Overall (at the flow diagram level)
- Component level

Cyclomatic complexity of graphs is known to yield good insights about the complexity of the graph, and the Xsemble flow diagram may be analyzed to understand the overall complexity of the domain, and that of a part of it.

While LOC (Lines of Code) is not a good measure to understanding the size of a large program, it may still be applicable to tiny components to assess their relative sizes. Further, the static code analyzers may be used to assess the complexity of the components.

By default, Xsemble components are meant to be tiny. So, a component having a large LOC or high complexity (as against its size expected) is not desirable. Adequate automated alarms may be developed to report crossing these thresholds.

4.10. Dependency Analysis

In Xsemble, the dependencies are explicitly mentioned against each component. This makes it extremely easy to upgrade a library or to change it. This can also be useful from the license management perspective, where licenses of certain libraries may be unacceptable and so the use of those libraries needs to be avoided.

5. Conclusion

IT projects are regarded as being inherently complex to manage. However, the Xsemble technology directly attacks the very pain areas which make it difficult. It delivers artifacts and data that make it amenable to measure and control. Combined with traditional tools and techniques, we get an end-to-end process that is smooth and crisp. This process is compatible with both waterfall and iterative (agile) methodologies. 10 specific ways in which the the new information may be used for better project management are covered.