Value Stream Mapping and Pull System for Improving Productivity and Quality in Software Development Projects

H. K. Raju¹, and Y. T. Krishnegowda²
¹University of Mysore, Mysore, India
Email: rajuhk2003@yahoo.com
²Maharaja Institute of Technology, Mysore, India
Email: ytk_gowda@yahoo.com

Abstract—In today’s business environment, increased globalization, rapid technological advances, and other competitive pressures are influencing the way the software companies used to manage software development projects. Software companies are competing to bring in more efficient process to improve productivity and quality in software development. In software industry, therefore, there has been a significant focus shift from traditional waterfall model to iterative agile/scrum methods to continuous lean software development in the last several years. The traditional waterfall model is the dominant project management paradigm. Agile methodology is direct response to waterfall model, and lean is recent one. In practice, when software companies make an attempt to adopt agile/scrum methods or bring in any process improvement, the team is unwilling to alter their way of doing day-to-day work. This mismatch between processes makes the team very difficult to accept the new or improved approach and at the end regress to their former process – waterfall model, with confusion. On the other hand, flow based pull system (or Kanban) offers an opportunity to build on what is already working and gradually makes the transformation to better process. And the Value Stream Mapping (VSM) helps to bring in significant improvement by identifying and eliminating/reducing non-value-add activities during software development. Value stream mapping and pull system (or Kanban) are based on lean principles. Pull system (or Kanban) integrated with value stream mapping can positively impact the productivity and quality in software development projects.

Index Terms—agile methodologies, function points, kanban, pull system, scrum, software development, value stream mapping, visualization, waterfall model, workflow, work-in-progress

I. INTRODUCTION

Software applications are the driving force of modern business operations [1]. Software development has been an important but troublesome process. In software development projects, software professionals are mostly dealing with the technical aspects of software development activities. The management aspect of software development process is ignored to the largest extent. But, in today’s marketplace, increased globalization, constant technological advances, and other competitive pressures are influencing the way the software companies used to manage software development projects. Software companies are working
towards bringing in more efficient process to improve productivity and quality in the software development process. The management aspect of software development is gaining more prominence these days. When we look at the global business trend from late 1990s, a massive change has occurred in the business world as a result of information technology available to everyone. The business world has irrevocably impacted by computerization, the Internet, the Web, and the range of inexpensive information processing and transmission capabilities. Information characterized by speed, accessibility, and most of all relevancy became the norm. The Internet and the Web have become a very common and very economical way to carry out business transactions. Thus, there is a fundamental shift in the way we used to carry out the business. Driven by this revolution in the business world, global economy has emerged [2]. Thus, large changes in all walks of life and profession have resulted from the advent of information technology. These phenomenal changes have led to a huge demand for more complex, integrated, robust, and resilient software systems.

In today’s global economy, software industry is the most strategic industry. Software development creates new opportunities for business change by providing business solutions. The resulting transformation in the business environment place large demands on software development capabilities – productivity and quality. Changes to existing business that result from the introduction of new software solutions eventually reach a level of major transformation. The transformed business also increases demand on software development capabilities because business leaders and decision-makers expect software industry to produce new applications that support the business in its transitional state. And, once the business successfully completes the transformation, it has new demand for software development. Thus, the software industry today has evolved as one of the most strategic industry. And, experts assess that, this trend will last at least another 50 years [3].

On the other hand, the software development process has more uncertainties and complexities as compared to manufacturing process. Software product is not rolling out as a physical and tangible product. Instead, it is creating an intellectual property, which is highly dependent on software professionals’ innovative thinking, creativity and efficiency [4].

In addition to this, today’s software development environment is geographically distributed – multi-location distributed teams with different time-zones. The development teams have to respond quickly to rapidly changing customer requirements and priorities (between 30 and 50 percent of all features are unnecessary and add overhead). And the team members are assigned multiple projects and are always overloaded with work. This leads to the situation that a large portion of software development problems are rooted in their natural workflows [5]. This makes the day-to-day activities clogged and thus gradually deteriorating quality and productivity in software development projects. Software community is looking for faster delivery with focus on creating highest value to the customer. Thus, there has been a significant focus shift from sequential waterfall model to iterative agile/scrum methods.

In practice, when software companies make an attempt to adopt agile methods, the team is unwilling to alter their way of doing day-to-day activities. The mismatch between the processes makes the team very difficult to accept the new process and ultimately regress to their former process [6]. In this situation, pull system (or Kanban) offers an opportunity to start out with the current state of the process and proceed to bring gradual transformation to the new process in small steps. And the value stream mapping helps to bring in greater efficiency to the software development process by uncovering the non-value-added activities and help deliver value to customers. Pull system (or Kanban) along with value stream mapping is a method for bringing incremental and evolutionary changes in software development activities [7]. This method limits work in progress (WIP), identifies unproductive work and helps improve productivity and quality in the software development process.

The objective of this paper is to provide an understanding and insight into value stream mapping and pull system to software development process. We also discuss a case study where the pull system (or Kanban) integrated with value stream mapping has been implemented in four phases to improve productivity and quality in the software development project.

II. OVERVIEW OF CURRENT SOFTWARE DEVELOPMENT MODELS

In software industry, currently, two models dominating the software development activities – waterfall model and agile/scrum model. Waterfall model is sequential development model. It is the predominant model in the software development community. It is there since 1970s. Agile/Scrum model is iterative development model. And, it is direct response to traditional waterfall model. According to a 2010 survey by Forrester, only
about 35% of software companies surveyed states that their development process is based on agile and lean principles [8]. Therefore, waterfall model is still a dominant software development model in the industry today.

A. Waterfall Model – Sequential Approach

The sequential approach to software development process is termed as waterfall model, after a model described by Dr. Winston W. Royce in the mid 1970s [9]. In waterfall model the work progresses in sequential phases. These phases typically include requirement analysis, high level design, low level design, coding and unit testing, integration testing, system testing, and release. The documentation involved in this model is quite a lot, as documents developed in the previous phase needs to be signed off before proceeds to the next phase. A typical waterfall model is shown in Fig. 1.

In this model, the project team do all the requirement analysis in one go, sign off, then all the design in one go, sign off, then all the coding in one go, sign off, and then finally, the complete system testing in one go. Then release. The major problem with the waterfall model is its inflexibility. It is especially ineffective and inefficient in responding to changing customer requirements. In waterfall model, testing normally takes place only after coding is complete. Unfortunately, most of the time the development work continues throughout the cycle while testing starts only late and never seems to get enough time to complete all testing activities. According to Royce, this model is risky and invites failure.

The waterfall model rarely works because it is predicted on stable requirements. In today’s marketplace, the customer requirements and priorities are rapidly changing. This approach takes several months to years to deliver a project.

In order to address some of the issues in waterfall model, the software industry adopted V-Model. This model is an extension of waterfall model and is based on association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle there is a directly associated testing phase. On the left side, development activities including requirement analysis, feature specifications, high-level design, and low-level design proceed from top to bottom. On the right side, testing activities, including, unit testing, integration testing, system testing, and acceptance testing are completed in a bottom-up fashion. In this model, test plans are developed along with each development activities, and the tests will executed in reverse order after coding is completed. Coding phase joins the two sides of the V – Model [10]. A typical V – model is shown in Fig. 2. The advantage of this model is that the test plans can be completed earlier in the process, resulting in shortened overall development and testing time.

![Figure 1. A typical waterfall model.](image1)

![Figure 2. A typical V – model.](image2)
B. Agile Method – Iterative Approach

Agile method is basically an iterative approach to software development. It represents a major departure from traditional waterfall model. *Agile manifesto* was written by experienced practitioners and was formalized in 2001. According to the *agile manifesto*, agile software development is based on four core values [11]:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

There are many agile software development methods. These include – Dynamic System Development Methods (DSDM), Extreme Programming, Crystal Clear, Feature Driven Development, Scrum, and others. Of all the agile methods, Scrum is very popular these days. Scrum method was developed by Ken Schwaber and Jeff Sutherland in 1990s. Scrum is unique because it uses the real-time progress of a project, where feedback loops constitute the core element, to plan and deliver the application. In Scrum, unlike waterfall model, we use smaller batch sizes and short delivery cycles called *sprints*. These sprints are typically one to four weeks in duration (we used four weeks sprint in our projects).

In each sprint, we do a little bit of requirement analysis, a little bit of design, a little bit of development, and a little bit of testing in order to deliver a handful of prioritized features every few weeks. The team members coordinate their work in daily stand-up meetings, which normally lasts for about 15 minutes. At the end of each sprint, product owner and team members meet to assess the progress of a project and plan its next sprint release. This allows the project manager to adjust and revise the project plan based on completed work, not predictions. This results in enormous improvements in quality work, delivery time, and customer satisfaction.

A typical scrum model is shown in Fig. 3.

![Figure 3. A typical scrum model.](image)

Though scrum is a better fit than waterfall model to software development, it has pitfalls when it comes to reality. The quality of requirement analysis suffers as team members rush to prepare for upcoming sprints, the quality of current development activities suffer when busy team members are unable to respond to issues during the iteration, and finally the quality of the product itself suffers as the development work continuous throughout a iteration while testing starts late at the very end of the iteration and never seems to get enough time.

III. INTRODUCTION TO VALUE STREAM MAPPING

For any organizations, there are THREE types of activities are undertaken, as shown in Fig. 4:

- **Value-add activity** – add value to the product/services. And customers are willing to pay for it
- **Business non-value-add activity** – wasteful but are necessary for running the organization. And customers are unwilling to pay for it
- **Non-value-add activity** – involve unnecessary actions not adding any value to the product/services. Absolutely waste. And customers are unwilling to pay for it

The first step in process improvement program starts with examining the process from the customer’s perspective – both the internal customer

![Figure 4. Three types of activities.](image)
at the next steps in the software development and the external customer/end-user [12]. Identifying what the customer wants from the process defines the value. Through the customer’s eyes, you can observe a process and separate the value-added steps from the non-value-added steps. In general, only a small number of the steps add value to the product, as far as the customer is concerned.

Value stream is the set of activities that convert customer needs into delivered product and services [13]. It is the stream (flow) of value added activities in the process.

Value Stream mapping is well understood in manufacturing sector. According to Michael Pitcher (president, Operations Excellence Consulting, Inc.), Value stream mapping is a lean tool that employs a flow diagram documenting in high detail every step of a process. It is a fundamental tool to identify waste, reduce process cycle times, and implement process improvement [14].

Value stream mapping help bring about greater efficiency to organization, improvement to processes and methods, and most importantly, deliver value to customers. Value stream mapping can help both identify and eliminate/reduce non-value-add in any organization.

IV. VALUE STREAM MAPPING IN SOFTWARE DEVELOPMENT PROCESS

In software industry, the work is always organized around projects. Software projects vary widely in complexity, technology, size, number of people involved, and lead time. All projects are unique and different. But if you start with the customer, define value, and then map the process that adds value to the customer, identifying workflow can be more manageable.

Look at the software development activities in any software development company. Software engineers are all very busy sitting in front of lap-tops and computers, looking at technical specifications, designing and developing software product, and having series of long meetings with the team members. It is very difficult to measure software professionals’ value-add productivity by looking at what he or she is doing. For this, we have to follow the process and progress of actual software project the software engineers are working on. During software development activities, identifying the source of waste very challenging. In this situation, value stream mapping is very helpful in identifying waste to optimize the software development process.

Although value stream mapping is widely used in Lean manufacturing, its utility is comparatively unknown in the software development community. According to Ted Rivera of IBM, value stream map exists for two purposes – to identify and end wasteful activities, and to stop waste from happening in the first place [15].

Value Stream Mapping (VSM) for software development is the set of all the activities required to develop a software application and includes how it is conceived, developed, and released. It includes both value-added and non value-added activities. VSM helps to understand the current process to analyze non value added (waste) activities during software development. It involves drawing the current state as well as the desired future state and a plan to get there. Though VSM has its root in Lean manufacturing and the Toyota Production System, it has shown to be powerful in improving software development processes. VSM integrating with the Pull System (Kanban) not only shows the presence of non value added (waste) activities but also lets the development team know where exactly is the problem in the workflow with a goal of improving the productivity and quality in software development Process.

Here is an example (Fig. 5) of a VSM for a typical waterfall model to software development process. Notice that how the Value Add activities tend to be fairly short bursts followed by longer delays as the process winds through common governance “gates”. The overall efficiency is 30%, meaning 70% of the activities involved are wasteful, mostly in the form of delays, prioritization, reprioritization, handoffs, workarounds, duplication, documentation, etc. This illustrates an important point about how the VSM can expose opportunities for improvement.

![Value stream in a typical waterfall model](image)

Figure 5. Value stream in a typical waterfall model.
For software development process, the value stream mapping begins with identifying the current steps in the process. The current value stream map shows work processes as they currently exist [16]. Once the each phase is identified in the current value stream mapping, the number of activities (in detail) in each stage is identified. Calculate the work time and waste time for each step by identifying how long each step is going to take. Then total work time and waste time for the entire process is calculated. Here the team has got lot of opportunities to identify the value add and non value add activities. Then more efficient and waste-free value steam can be tailored. The value stream mapping is the first step to transform to pull system. It is a very efficient tool.

For creating a Value Stream Map (VSM), we are actually mapping the flow of work for a particular software development project. We are not mapping the amount of time people are working on different projects. Following are the steps to create a Value Stream map:

a) Identify the actions that take place
b) Specify how much of the time these actions were being worked on actual work was taking place and how much time was spent on wasting
c) Look, and denote any loop backs present in the workflow
d) Total up the average time working on the project
e) Analyze the entire value stream against value-added and non-value-added activities.
f) Develop and initial value stream map for pull system (or Kanban)

V. INTRODUCTION TO PULL SYSTEM (OR KANBAN)

In manufacturing system, there are several departments are set up. These departments produce different types of products. Actually, the same equipments are used to produce different types of products. The equipment changeover is done for making different types of products with the same equipment. Here, the individual departments will try to minimize this equipment changeover. As a result, a specific department produces large-volume items before it makes equipment changeover. Since each department is making what it wants, there will not be any real coordination between departments. To keep the downstream departments busy, there will be huge inventory buffers between departments. So the departments working according to independent schedules will be pushing materials into these inventory buffers.

In the 1950s, Toyota’s leaders took a study tour to US plants to understand their manufacturing progress. But, they are not convinced by the system as the manufacturing process had many inherent flaws. During their stay in the US, Taiichi Ohno and his associates were fascinated by importance and influence of supermarket to daily life in America. They are inspired by how well-run supermarkets replenish the shelf, once the items are reduced to lowest level prescribed.

Ohno observed that, the inventory was necessary to allow for smooth flow. But, individual departments building products/parts to a schedule using push system would naturally overproduce and creates large banks of inventory. On the other hand, a pure one piece-flow system would be a zero inventory system where products/parts just appear when they are needed by the customer. It is an ideal case.

As a compromise between the ideal one piece flow and push system, Ohno created small store of parts between operations to control the inventory. When the customer takes away the specific items, they are replenished. Since factories are large and spread out and suppliers of parts are a distance away, Ohno needed a way to signal that the assembly line had used the parts and needed more. He used simple signal to refill it – cards, empty bins, empty karts, called Kanban, with detailed information regarding a specific number of parts and location. Kanban is a Japanese term, means “visual card,” “signboard,” or “billboard. But it is taken more literally as a signal of some kind to trigger an activity. Toyota’s whole production system (known as TPS – Toyota Production System) is based on this pull approach. It is popularly known as the Kanban system. It ensures efficient flow and production of materials in a just-in-time production system. There is no more overproduction than the small amount on the shelf. And some direct connection has been established between what the customer wants and what the company produces.

The pull system (or Kanban) is extremely successful in manufacturing system. It is this pull system in manufacturing process is the inspiration behind what the software industry is talking about pull system (or Kanban) for software development process.

VI. PULL SYSTEM (OR KANBAN) FOR SOFTWARE DEVELOPMENT IN A NUTSHELL
Kanban brings incremental improvements to the current process. It gradually eliminates the whole idea of iterations or sprints. Through continuous delivery of high-priority features/functions, it focuses on creating a highly responsive team rather than fitting the software development into rigid time-boxed iterations.

Kanban approach to software development means that only software application with features/functions requested by the client will be developed. This approach help eliminate what is known as gold-plating, where developers include apparently sophisticated ‘bells and whistles’ that may only be interesting to developers, which have not been requested by client, and may never be used [17].

Pull system (or Kanban) for software development was developed by David J. Anderson in 2006. It is a method for making incremental and evolutionary improvements in the software development process. Pull approach limits the number of features/functions locked in work in progress (WIP) and the next work item will be taken only when the current prioritized feature/function is developed. This helps to balance the overall development activities by elimination the clog in the development cycle. As a result, pull system (or Kanban) method will help achieve lower cost, high quality, better productivity, and lesser lead time.

According to David Anderson, there are five core practices for successful implementation of Kanban pull system for software development:

- Visualize the workflow
- Limit work in progress (WIP)
- Measure and manage flow
- Make all process policies explicit
- Improve continuously based on facts and explicit policies

According to Hernik Kniberg the main components of the pull system (or Kanban) in software development process are [18]:

- Visualize the workflow
- Limit the work in progress (WIP)
- Measure the lead time

According to David Joyce of the BBC Worldwide, Kanban pull will support the team’s journey of process improvement and will help expose the best solution. It is about the right work is being done at the right time, rather than who is doing the work. The basic principles of Kanban pull for software development are:

- Limit work in progress (WIP)
- Pull value through (with WIP limit)
- Make it visible (use visual control)
- Increase throughput
- Fixed Kanban backlog
- Quality is embedded in (not inspected in)

The beauty of the pull system (or Kanban) is that it can be adopted immediately to the current state of the process. From there we can evolve in small steps by recognizing bottlenecks, waste, and variability that affect the process performance.

Through detailed literature review and with our own experience in setting up the pull system (or Kanban), we would like to recommend NINE powerful steps for implementing pull system (or Kanban) to software development projects.

- Map the value stream for the project
- Create visibility in both the work being done and the way the work is being done
- Reduce lead time by eliminating or reducing queues
- Reduce delays by limiting work-in-progress (WIP) to the capacity of the team
- Pull value through the value stream (with WIP limit)
- Measure and manage flow to improve cycle time
- Increase throughput progressively
- Use models to recognize improvement opportunities
- Make all process policies explicit

A. Map the value stream for the project

One of the biggest wastes during software development process is usually waiting for things to happen. Delays in starting certain phases, delays in staffing, delays due to excessive documentation, delays in reviews...
and approvals, delays in testing, and delays in build release, etc. Mapping value stream is a good way to start discovering the waste during software development process. It visualizes the software development process showing productive times and waste/wait time. Identifying what the end-user/customer wants from the process defines the value for the software projects. The value stream is essentially all of the work that takes place during the software development. Therefore, identify all the steps in the project in order to develop the software application. Calculate the productive time and waste/wait time for each step by identifying how long each step is going to take. Then total work time and waste time for the entire process is calculated. This provides a starting point for evaluating productive and non-productive activities that take place during the software development process. Based on this assessment, develop an initial value stream map for improving the current software development process to deliver value to the customer.

B. Create visibility in both the work being done and the way the work is being done

Kanban screen is the core of visualizing the workflow. According to Anderson, posting different stages of software development in the workflow and visualizing the status of each work item will bring more transparency into the team. Team members can have the visibility into any work item. They can see what others are working on. They can see if they could help completing the existing work item before pulling a new work. Visualizing the workflow enable team members to self-organize among themselves. This leads to greater trust among team members. Team members can identify bottlenecks and take steps to resolve them in the flow. Furthermore, visibility can help managers make good decisions [19].

C. Reduce lead time by eliminating or reducing queues

During software development, work items spend a lot of time in queues – like waiting to be reviewed, waiting for documentation update, waiting to be signed off, handoff to next stage, etc. It is this queue that creates a bottleneck in the workflow and affects the overall lead time. A quick way to reduce the lead time is check for these queues and figure out a way to eliminate or reduce them. Or set limits on the amount of work that the team takes on at one time – stop upstream activities until capacity is available downstream [20].

D. Reduce delays by limiting work-in-progress (WIP) to the capacity of the team

Pull system (or Kanban) does not limit the time needed to complete work item, rather it enforces to set limitations on WIP. WIP Limit is a limit that governs the number of work items that can be in that state at any instant. Kanban uses these work in progress limits to change the basic project management paradigm from pushing work downstream to pulling work from upstream. If the state is below its limit, it may take possession of the work item from upstream. If a state is at its limit, it must wait for one of its items to be completed, and pulled from downstream.

E. Pull value through the value stream (with WIP limit)

According to the Standish Group, about 45% of the features of software applications were never used [21].

From this statistics (Fig. 6), most of the features/functions in any software applications don’t add any value to the customer. These features are unnecessary. They add overhead – create more defects, complexity into the software application, and thereby increases unnecessary cost to the company. Then, according to Corey Ladas, don’t build features that nobody needs now; don’t write more specs than you can code; don’t write more code than you can test; and don’t test more code than you can deploy [22].

F. Measure and manage flow in order to improve cycle time

Cycle time is the time measured when the actual work begins on the request and ends with the status ready for delivery. On the other hand, the lead time is measured from the time when the request is made and ends at
delivery. Lead time is more important from the customers’ point of view, while cycle time is the way of measuring the process capability by the developers. Lead time is much longer than the cycle time.

G. Increase throughput progressively

Throughput is the number of work items completed per iteration. It is the rate of delivery of customer valued features/functions into the software application. Two major variables regulate the throughput are – WIP and Cycle Time (this is measured from when you start working on the feature). This throughput allows forecasting of future process capability and set the new benchmark for the organization.

H. Use models to recognize improvement opportunities

Kanban encourages collaborative improvement of current processes using a scientific approach. This means that everybody on the team should be part of process improvement programs. A mere trial and error method should be avoided and more structured and organized methods should be used, like Lean, Six Sigma, Theory of Constraints, and other models.

I. Make all process policies explicit

In general, the software development teams have implicit policies about how different types of task are to be handled. Use process policies to define the basis of work. This will control the flow of work and help team members validate their understanding about what to do. Unless the process is understood and apparent to its users, it is very difficult to discuss its improvements.

According to David Anderson, without explicit process policies, any discussion of problems/process improvements tends to be emotional and subjective. Making process policies explicit, it is possible to hold rational and objective discussion to resolve the issues and identify opportunities for process improvements. These explicit process policies will help team to self-organize themselves and manage the workflow efficiently.

VII. A CASE STUDY OF VALUE STREAM MAPPING AND PULL SYSTEM FOR SOFTWARE DEVELOPMENT

This case study is taken from one of the projects that adopted value stream map and Kanban pull system successfully, as part of process improvement program at ADC Telecommunications. ADC Telecommunications, head quartered in the USA, has a Software Development Center in India. This center has been providing solutions for telecom sector. It has a strong and dedicated team of about 30 software professionals/engineers – design, development, and testing (manual as well as automation testing). In spite of this, the time to resolve the problem from problem reporting was about 10-14 weeks. Even then, the bug fixes were often not as expected and the new issues getting added into the bug list, taking yet another 6-8 weeks to respond. Number of bugs in the bug tracking system was growing day-by-day. The team adopted the waterfall model for its cutting-edge software development activities. Still it was taking several months (sometimes over a year) to make a new release to the market.

As a part of process improvement program, the management decided to adopt iterative method for the development process. As a result, Scrum has been introduced with four weeks sprint. Standard elements like cross-functional teams, daily stand-ups for about 15 minutes, and sprint planning have been implemented. The day-to-day status was made available and visible on a big black board – in front of which daily stand-ups have been decided to take place. After three to four iterations, the end result was so bad that the team in confusion messed up the entire thing. Daily stand-ups itself become a frustrating experience as 15 minutes extends up to 45 minutes to an hour. Daily fire fighting situations – almost everyday people worked late night. Nobody cared the big black status board but the engineer who is updating that on a day-to-day basis. Because of the sudden change in the process from sequential to iterative; the team is unwilling to alter their way of doing the day-to-day work. The mismatch between the processes made the team very difficult to accept the new process. And ultimately with the confusion they are falling back to their previous processes, unknowingly.

After the detail analysis of the situation Kanban pull system was selected to bring in incremental improvements to the process. The team developed current state value stream map. Analyzed value-added and non-value-added activities from the customer’s perspectives – both internal and external customers. Developed initial improved value stream map. Then, using This value stream map, the Kanban pull system was tailored and implemented. A real-time screen for Kanban pull system was set up with the open source software (OSS) application. To evaluate the effectiveness of this pull system, the team continuously
monitored the work items to flow through the different parts of the value stream. The team tracked the productivity and quality through the following performance data:

- Development time for prioritized features/functions
- Build release frequency
- Number of defects remain open per week
- Continuous improvement over iterations

The implementation was carried out in the following FOUR phases:

- Phase I – Develop a Value Stream Map (VSM)
- Phase II – Set up the Kanban screen to implement pull system
- Phase III – Link up the Kanban pull system with software measurement.
- Phase IV – Link up the Kanban pull system with project schedule

A. Phase-I: Develop an Initial Value Stream Map for Pull System

In this phase the team developed a value stream map in order to analyze how the value is being delivered to customers. They identified all steps in the process and created box for each step. Then the relationship between the steps was identified to represent the current workflow for the software development project. For each step (represented by box), the team listed down the activities to be carried out to complete that specific step. This activity was carried out by discussing with project team members and other key stakeholders of the project. With detail activities on hand, the team calculated how long each step is going to take – by collecting information on how many people are involved in each step, identifying working time required to complete the step, time wasted (non-productive work). The time wasted also includes the wait time between each step in the process.


During the analysis of the current value stream map, the following non-value-add activities are identified:

- Too many sub groups headed by project managers for handling the project (no coordination)
- Unnecessary reviews (just to follow some standards)
- Unnecessary approvals/sign-off (just because of the practice)
- Bottlenecks caused by un-prioritized activities
- Redundancy in system testing (test automation team and manual testing team)

Based on the analysis, the current state is expanded to improved value stream map by eliminating non-value-add activities, elimination of waste, improving the communication and information flow in the team. With this initial value stream map (Fig. 7), the real-time screen for basic pull system (or Kanban) is set up.

B. Phase-II: Real-time screen for basic Kanban pull system

In this phase the team adopted effective work breakdown structure [23] in order to develop appropriate work items for given features/functions. Kanban method is more concerned about the right work being done at the right time in the right way. They identified and organized all tasks to be monitored in the work in the most efficient way. Then real-time Kanban screen was designed and implemented to keep track of the work items. This screen includes the components like work items, team members, different stages, color code for status. The work items are placed in queues and prioritized work items are pulled by the respective team members/group to work on. These work items move through the workflow set on the screen. As soon as the task is completed at one stage, the
respective team member needs to update the status of the task before it proceeds to the next stage. The color code is assigned to each status – green (done), yellow (work in progress), and red (some problem in that stage and it needs attention from the project team). The model screen for the real-time pull system (or Kanban) is shown in Fig. 8.

For example, in the top row, a prioritized feature (feature 2) is broken down into three work items – W17, W18, and W19. These work items have to flow through four stages – Requirement Analysis, Design, Development, and Verification and Validation, the “Done” column display the status for each work item. In this case, W19 is completed successful, W17 is in progress, and W18 is in problem and needs attention from the project team. The “Release” column shows the overall status of the feature. In this case Feature 2 is in problem (color code red). And team members focus their attention to resolve the issue quickly. The group cannot pull new work from upstream. Similarly, Feature 4 is in progress. The group cannot pull new work from upstream. And Feature 1 is completed; this group can pull new work from upstream.

In this phase, the team gradually adopted the flow based model. The duration of the iteration is depending on the number of high priority features/functions to be released for that iteration (not fixed one).

### C. Phase-III: Kanban Screen with Function Points

In this phase, software measurement has been linked to the workflow. The software sizing and productivity based on experience with the Kanban workflow will help assess the performance of software development project and benchmarking productivity for continuous improvement. The team adopted Function Point Analysis (FPA). According to Software Productivity Research, software sizing with Function Points can help significantly improve the probability of completing a software projects successfully – on time and within

![Figure 9. Statistics of software projects done without and with FPA. (Source: Software Productivity Research)](image-url)
budget. Fig. 9 shows the statistics of software projects done without and with FPA [24]. The FPA technique quantifies the user functions that are meaningful to the software user. The Function Point measure relates directly to the business requirements, it can be applied throughout the software development cycle – from requirements analysis to release. Function Point sizing can provide a solid basis of estimate for cost and schedule plans at the beginning of the project [25]. Fig. 10 shows the real-time Kanban screen linked with Function Points. For example, the size of Feature 6 is of 10 Function Points, Feature 1 is of 8 Function points, and Feature 5 is of 5 Function points.

The Kanban flow with FPA, it is convenient to calculate the productivity – First Time Yield (FTY) for the iteration. The FTY for the iteration can be defined as the number of Function Points successfully completed divided by the total number of Function Points planned for that iteration. In this case, the workflow in the software development for four iterations can be visualized as shown in Fig. 11. For iteration 1, we have planned 95 Function Points. Calculate yield at each step – requirement analysis through testing, and multiply these together to get First Time Yield (FTY) for that iteration.

\[
\text{FTY (Iteration 1)} = Y_{11} \times Y_{12} \times Y_{13} \times Y_{14}
\]

\[
\text{FTY (Iteration 1)} = 0.96 \times 0.97 \times 0.84 \times 0.78
\]

\[
\text{FTY (Iteration 1)} = 0.61
\]

Likewise, for iteration 2, we have planned 80 Function Points, and the yield is 0.58. For iteration 3, we have planned 96 Function Points, the yield is 0.83. And for iteration 4 we have planned 36 Function Points, the yield is 0.59. Then the mean First Time Yield (Mean FTY) over four iterations is given by

\[
\text{Mean FTY} = (Y_1 + Y_2 + Y_3 + Y_4)/4
\]

\[
\text{Mean FTY} = (0.61 + 0.58 + 0.83 + 0.59)/4
\]

\[
\text{Mean FTY} = 0.6525 \approx 65\%
\]
This means Yield = 65% and Rework = 35%
With this, we analyzed root causes for 35% rework. The team worked towards reducing the rework. After 6-8 iterations, the rework got reduced by 14% to 21%. Subsequently there is an improvement in the cycle time for the new release [26].

D. Phase-VI: Kanban Screen with project schedule

In the first cut, the team successfully adopted the Kanban method to initiate the basic workflow. Once the entire team became comfortable with this system, in the next phase, Software measurement was linked to the Kanban workflow. The team adopted FPA to monitor the performance of software development project. In this fourth phase, project schedule was integrated into the pull system (or Kanban), as shown in the Fig. 12.

Normally, software engineers don’t have patience to give effort estimation methodically. They ignore this part, and are comfortable with “gut feeling + buffer” methodology. Therefore the estimation is frequently done irrationally. The third phase exercise really helped us to try out effort estimation methodically. Furthermore, this helped us to allocate the resources optimally to deliver value to the customer. And created a smooth continuous flow of the software development. This resulted in setting up the trend for disciplined software development in the organization.

E. Realized benefits of Kanban pull system integrating with value stream for software development process

Apart from successful transition from rigid waterfall model to flexible flow-based model, the gradual implementation of Kanban pull system resulted in the following benefits:

- The workflow is based on the value stream map developed to deliver value to the customer. The team tracked the productivity and quality through this workflow.
- The color code in the workflow was introduced to display the real-time status of the software development activities.
- With the PULL approach, the work-in-progress (WIP) that was allowed to enter the development process was significantly reduced to ensure that the workload was not beyond the capacity of the project team.
- Non-value-added activities like working on incorrect, incomplete, or misunderstood requirements were significantly reduced.
• Increased productivity – significant improvement was made in the cycle time and frequency of build releases.
• Improved quality – rework got reduced from 35% to 21%
• The number of defects reported open and the number of defects fixed each week showed considerable improvements.
• Better information flow – real-time updates were made available to all team members
• Reduced cost – helped to allocate optimum resources to deliver value to the customer
• Flexible – helped prioritizing and re-prioritizing requirements throughout the development process
• Avoided over-production – limits on work in progress changed the traditional software development paradigm from pushing work downstream to pulling work from upstream
• The transparency in the workflow increased trust among team members

CONCLUSIONS
In software industry, it is very common during any process improvement programs that the people are unwilling to alter their way of dealing with day-to-day activities. Finding problems and developing more efficient process is a challenge. In this situation, the value stream map, and the Kanban pull approach are found to be more efficient and effective. The first step is to create a current value stream map to analyze and segregate value-added and the possible sources of non-value-added activities during software development process. Then develop an improved value stream map. Based on this value stream, tailor the Kanban pull system.
Kanban pull integrated with value stream mapping focuses the team’s energies on improving those areas in the value stream that are truly constraining other work, and thus harming workflow through the value stream. It helps avoid wasting/waiting time on activities that do not really add value to the customer. By having a before, during, and after action method of making small changes, the project team learns what works and what does not work to develop an efficient process. Thus, the Kanban pull system along with the value stream mapping can help to start out the process improvement with where we are and proceed to bring incremental and evolutionary changes in software development activities.

ACKNOWLEDGMENT
The authors would like to thank Maharaja Research Foundation (MRF) – University of Mysore, Mysore. This work was supported through the part of the grant to research work.

REFERENCES