**PROJECT PLAN ON**

**SINGLE PANE MANAGEMENT OF CLOUD EMAIL SECURITY**

**SUBMITTED BY**

**KUSH BAVISHI 1MS12CS049**



**M. S. Ramaiah Institute of Technology**

**(Autonomous Institute, Affiliated to VTU)**

**BANGALORE-560054**

**Department of Computer Science & Engineering**

***Under the guidance of***

|  |  |  |
| --- | --- | --- |
| **Suresh Gopathy**  Technical Leader  Cisco Systems, Inc.  Bangalore | & | **S. Rajarajeswari**  Assistant Professor  Department of Computer Science & Engineering |

**SCRUM MODEL**

Scrum is an iterative and incremental [agile software development](https://en.wikipedia.org/wiki/Agile_software_development) framework for managing product development. It defines "a flexible, [holistic](https://en.wikipedia.org/wiki/Holism) product development strategy where a development team works as a unit to reach a common goal", challenges assumptions of the "traditional, sequential approach" to product development, and enables teams to self-organize by encouraging physical co-location or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines in the project.

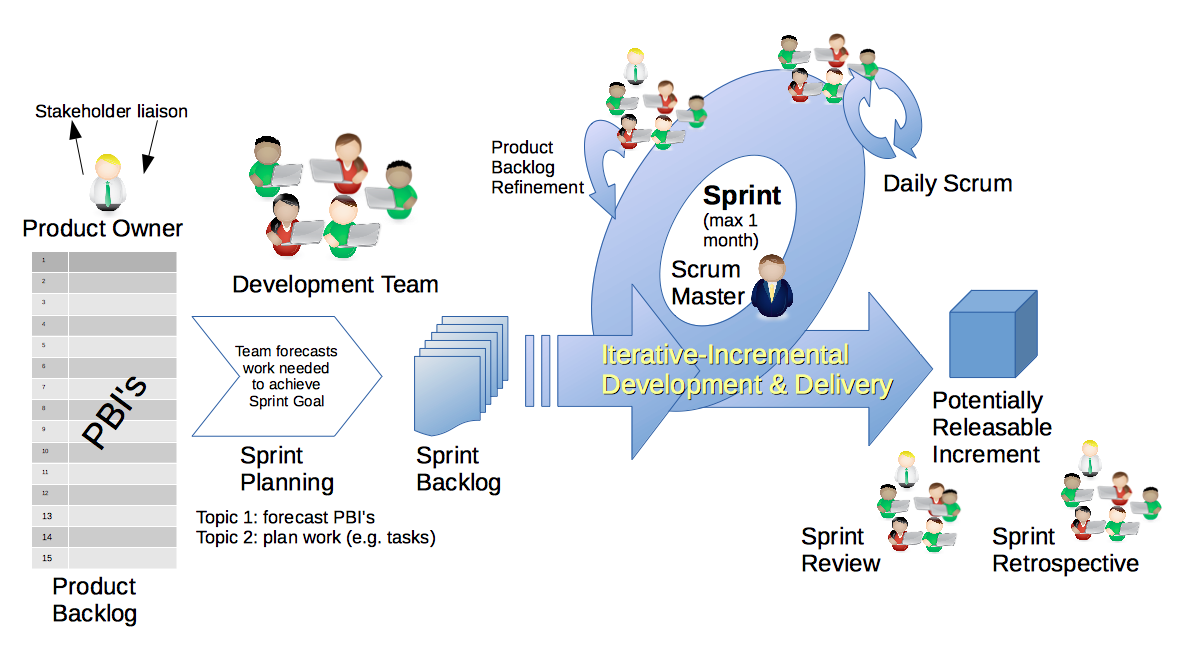
A key principle of scrum is its recognition that during production processes, the customers can change their minds about what they want and need (often called requirements volatility), and that unpredicted challenges cannot be easily addressed in a traditional predictive or planned manner. As such, scrum adopts an [empirical](https://en.wikipedia.org/wiki/Empirical) approach—accepting that the problem cannot be fully understood or defined, focusing instead on maximizing the team's ability to deliver quickly, to respond to emerging requirements and to adapt to evolving technologies and changes in market conditions.

**Workflow**

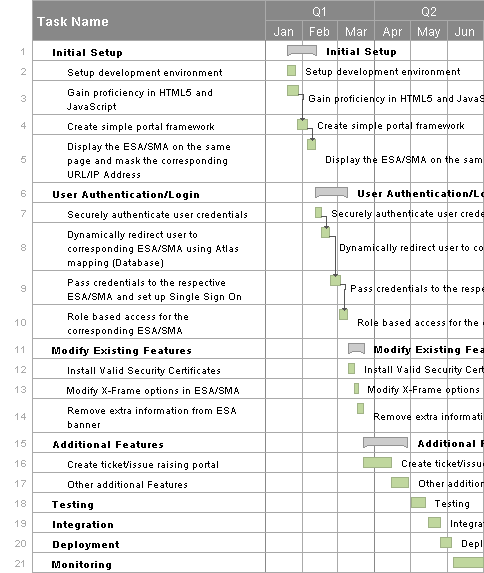
A sprint (or iteration) is the basic unit of development in scrum. The sprint is a [*time boxed*](https://en.wikipedia.org/wiki/Timeboxing) effort; that is, it is restricted to a specific duration. The duration is fixed in advance for each sprint and is normally between one week and one month, with two weeks being the most common.

Each sprint starts with a sprint planning event that aims to define a sprint backlog, identify the work for the sprint, and make an estimated commitment for the sprint goal. Each sprint ends with a sprint review and sprint retrospective, that reviews progress to show to stakeholders and identify lessons and improvements for the next sprints.

Scrum emphasizes working product at the end of the sprint that is really *done*. In the case of software, this likely includes that the software has been integrated, fully tested, end-user documented, and is potentially shippable.



**PROJECT SCHEDULE USING GANTT CHART**

****

**TOP 5 RISKS AND MITIGATION STEPS**

## 1. Inherent Schedule Flaws

Software development, given the intangible nature and uniqueness of software, is inherently difficult to estimate and schedule.

**Mitigation**: On agile projects the team is heavily involved in planning and estimating through activities such as XP's planning game and Wideband Delphi workshops. By working in short increments the true velocity of the team quickly emerges and is visible to all stakeholders who are now more closely involved in the project.

**2. Budget Risk**

A project poses budget risk if there is:

* Wrong estimation of cost
* Cost overrun
* Project scope expansion

**Mitigation:** To undertake better effort estimation. Building a COCOMO model can mitigate the budget risk. Build an SRS model.

## 3. Requirements Inflation

As the project progresses more and more features that were not identified at the beginning of the project emerge that threaten estimates and timelines.

**Mitigation:** Agile projects plan in the regular trade-off discussions about features and estimates at every iteration boundary. Changes and requirements inflation are accepted as a fact of software projects. Rather than utilizing change-suppression mechanisms, prioritization sessions are scheduled that allow worthwhile changes to proceed and initially envisioned features to be superseded if the business gives their authorization. It has never been possible to squeeze a pint into a quart cup, but now at least we anticipate the likely issue and have mechanisms in place to address the matter as part of the project from its early stages.

**4. Operational Risk**

Risks of loss due to improper process implementation, failed system or some external events risks.  
Causes of Operational risks:

* Failure to address priority conflicts
* Failure to resolve the responsibilities
* Insufficient resources
* No proper subject training
* No resource planning
* No communication in team.

**Mitigation**: Assign project to the person having skills required for it, Continuous training of skills, Communication between teams.

**5. Technical risks:**

Technical risks generally leads to failure of functionality and performance.  
Causes of technical risks are:

* Continuous changing requirements
* No advanced technology available or the existing technology is in initial stages.
* Product is complex to implement.
* Difficult project modules integration.

**Mitigation**: Using of SDLC such as Agile which can overcome the issue of continuous changing requirements, breaking the product to make it less complex.