Lead SD, Kanban, XP, DevOps, and Scrum.

Scrum

Scrum can be called the framework of agile software development. Alesia Krush shared an experience: "once I met a friend and told him that my new job was about" agile ". One of his questions was, are you doing something like scrum every day?". In many people's opinion, Scrum is synonymous with agile.

First, Scrum is a management framework. Scrum clearly defines a model according to which developers plan work, update plans and analyze processes. The framework introduces the role of Scrum master, which is a role dedicated to facilitating the process and ensuring compliance with the process.

Artifact

The main "artifacts" of Scrum (information disseminators) are:

1. User story. A small part of the function is that the team will work in a time period called sprint. The usual format is: as a [user role], I want [the system to do this and that] in order to [deliver this and this business value]. It must have a "completed definition" to determine whether the story has been executed correctly.

2. Task. It can be related or unrelated to user stories. For example, setting up a new development environment or studying CPU memory problems are tasks that have nothing to do with user stories.

3. Backlog. A list of user stories and future sprint tasks.

4. Sprint backlog. Select user stories and task lists (also known as "work items") from the current sprint in the backlog.

5. Product increment. A potential deliverable function block delivered at the end of sprint.

6. Extensions. Reports such as burndown chart and velocity are used to track the progress of the team.

role

1. Development team. It includes developers, QA engineers, UI / UX designers, business analysts, and others as needed. Scrum teams usually have 3 to 9 members. When 9 people are not enough, the team is divided into two.

2. Scrum master. Preside over daily scrum meetings, planning / update / review meetings, and help team members solve communication problems. Scrum master is not a team member, so they can work with multiple teams at the same time.

3. Product owners. Representatives of stakeholders communicate the vision of the scrum team (as the basis for user stories) to the scrum team, give priority to user stories at the end of each sprint, and accept or reject them.

value

1. Commitment (achieving goals in Sprint).

2. Courage (do what you think is right).

3. Focus (work items in the current sprint).

4. Openness (on any challenges).

5. Respect (trust other people's abilities).

Kanban

The Kanban framework was invented by Toyota Engineer Taiichi Ohno. In the late 1940s, Toyota representatives observed how supermarkets restock according to the goods on the shelves. This prompted Toyota to establish a supply system, and the production plan will be driven by actual consumption.

One of Kanban's key ideas is to avoid surplus. Kanban uses Kanban cards and Kanban boards to visualize the movement of resources in the production cycle. This enables everyone to understand the process to the greatest extent and helps managers solve surplus / shortage problems in real time.

Kanban system also introduces the concepts of "pull" and "push", so that workers can work according to their ability, rather than on the conveyor belt or in the form of to-do list.

In software engineering, Kanban means that the amount of work that can be carried out at one time is limited. In other words, there is an upper limit to the number of cards in the "in progress" column of kanban. This is to increase focus and reduce context switching.

Another aspect of Kanban development is that activities are always closely related to customer needs and maintain continuous communication with customers. Nothing will happen unless it is economically beneficial to customers.

principle

1. Focus - reduce multitasking;

2. Reduce waste;

3. Put customers' needs first (i.e. their business needs ROI);

practice

1. Visualization;

2. Limit work progress;

3. Process management (can be completed by managing queues or limiting work);

4. Clear policy;

5. Use feedback loop;

6. Experimental evolution;

The key difference between Kanban and scrum is that Kanban is continuous, while scrum is iterative. Kanban is more suitable for teams with a lot of unplanned work (support problems; emergency fixes; emergency function requests) during sprint. In this way, teams can reorder tasks at any time without waiting for the end of sprint.

Lean

Mary Poppendieck, a lean developer, has achieved great success in her career. She led her to write lean software together with Tom Poppendieck. Because lean borrowed a lot from Kanban, there are many similarities between the two methods.

Like Kanban, lean tries to avoid waste and maximize value for customers. Unlike Kanban, lean has some regulations on engineering practice (such as TDD). At the same time, lean is not so strict on delivery time, and the team may be ready to deploy at any time.

XP - Extreme Programming

Extreme programming began with an experiment by Kent Beck. His idea is to make the programming practice to the extreme and see what happens. For example, code review replaces code inspection. Later, as more and more companies began to adopt this method, such as daily integration testing, some strict rules began to be ignored.

Contrary to the traditional concept, XP is not only a simple equal pairing programming, but also provides a process management algorithm.

Another thing to note is that ideally, all XP operations should be used together, otherwise they will not work properly.

The management of XP has been criticized by some project managers. For example, persistent customers are considered to be a source of pressure. In addition, without any requirements and design, the system may be ineffective.

value

The XP value is related to the value in scrum. See the table:

Like Kanban and lean, XP also pays attention to waste.

1. Plan the game;

2. Test Driven Development ("write unit test first");

3. Pairing programming;

4. Team (actual users of the customer / program can be used for feedback);

5. Continuous integration;

6. Reconstruction design improvement;

7. Small version;

8. Coding standard;

9. Collective code ownership;

10. Simple design;

11. System metaphor (naming things in a way understood by programmers, customers and others);

12. Sustainability.