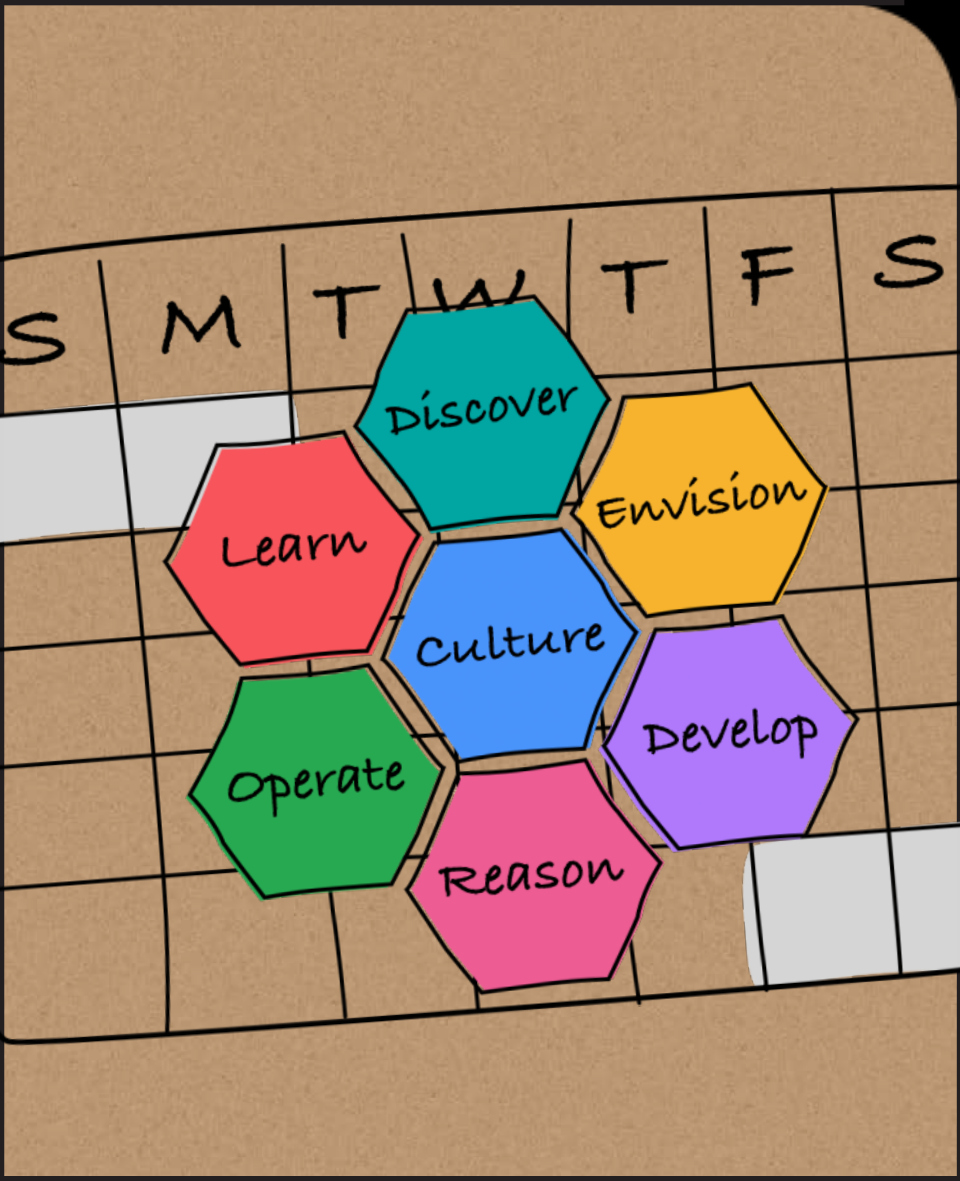


IBM Garage Day-to-Day

Field Guide

IBM

Download the current version of the IBM Garage Day-to-Day Field Guide

https://ibm.biz/ibm-garage-day-to-day-field-guide

What is the IBM Garage?

The IBM Garage™ is a bold and exciting way for IBM experts and clients to engage to solve challenging business problems and align on a common goal. The Garage brings together IBM domain experts, modern practices, and leading technologies to build solutions. Working together, you can prioritize, prove, and scale new solutions that deliver measurable results to your enterprise.

IMAGINE A BRAVE, NEW WAY TO WORK

Modernize the way you work. Embrace the IBM Garage Methodology, which brings together Lean Startup, design thinking, Agile, DevOps, site reliability engineering, and many other best practices. The practices in this approach focus on human needs and take an iterative approach to deliver the value that meets those needs.

Embark on a culture change. Culture is key to the success of your enterprise transformation. Focus on creating small, self-contained teams that are autonomous and able to make decisions that are based on efficiency and knowledge.

Take advantage of technology. Build your applications on the cloud and incorporate leading-edge technologies, which include AI, analytics, and edge computing. Use tools and services to streamline your processes.

What's inside?

This field guide provides a high-level overview of how to do day-to-day work in the IBM Garage using best practices and tools.

LEARN IT
A summary of the concepts.

GET STARTED
Tips to start your IBM Garage journey.

Start with Enterprise Design Thinking

When you work with IBM in the IBM Garage, Enterprise Design Thinking helps you focus on your users and their needs to deliver more useful, usable, and desirable solutions. To accomplish your goals, you must conceptualize, design, refine, and prioritize features that will delight your customers.

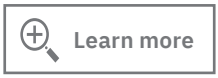
WOW YOUR USERS

Conduct workshops. Facilitated workshops align your team in the areas of business framing, technical discovery, design, and architecture using pre-planned activities. The outcome is a minimum viable product (MVP), wireframes, and stated goals and risks.

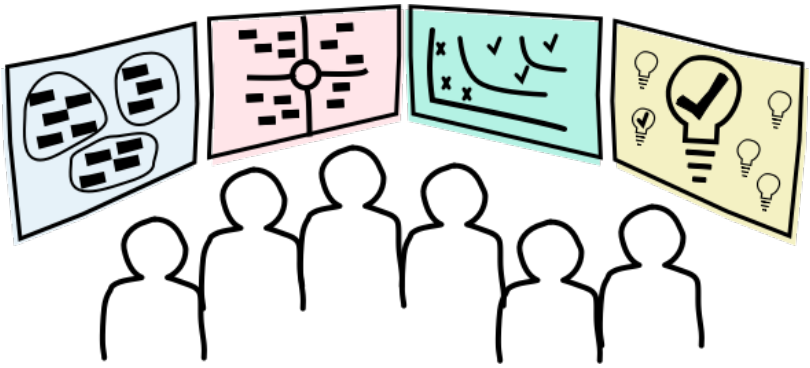
Define an MVP. An MVP is the absolute minimum function needed for your target persona to have a delightful experience while accomplishing a goal.

Create wireframes. Build wireframe sketches to communicate the basic aspects of the user interface and the most important task flows for the MVP. Use them to quickly gather feedback from your target audience and learn more about what matters to them.

Review outcomes. Be sure to review the workshop outcomes so that everyone agrees on the decisions and direction.



Learn about Enterprise Design Thinking.
https://www.ibm.com/garage/method/practices/think/enterprise-design-thinking/

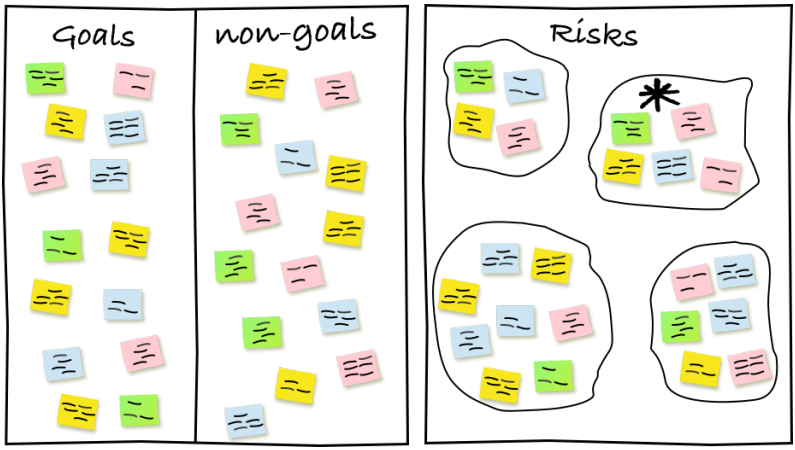


MVP statement: Sally can upload a photo of a pet and order a pair of socks customized with the pet's photo and name.

Inception

Take the outcome of an Enterprise Design Thinking workshop and turn it into something that can be built using Inception. The Inception workshop aligns the design, architecture, and development teams on project goals, risks, and scope. You also break the minimum viable product (MVP) into individual user stories and create a ranked backlog.

ALIGN ON GOALS, DRIVERS, RISKS, AND UPCOMING WORK



Learn more

Read about the Inception practice.
https://www.ibm.com/garage/method/practices/think/inception

Involve the whole delivery team. Include the product owner, architect, designers, and developers in the inception activities.

Consider the UX design. The designer might want to share early sketches with the team to help the team understand the vision.

Define goals and non-goals. Start with the goals from the MVP workshop and work through a guided exercise or brainstorming activity using sticky notes to build a concise list of goals and non-goals for the MVP. Goals can include design goals, functional goals, and technology goals. You can continue to refine the goals throughout the inception workshop.

Identify risks and possible mitigations. Revisit the risks defined in the MVP workshop. During inception, you should capture risks you missed, and focus on the most significant technical and project specific risks. Discuss and capture possible mitigations for each identified risk.

Write user stories. Developing user stories keeps progress visible, allows for regular feedback, and keeps the focus on value. Start by breaking the MVP into user journeys, or activities. Then decompose the activities into small user stories that can be completed in about a day. Shoot to define a backlog of user stories for the next 1 – 2 weeks.

Writing user stories

During the inception workshop, your team distills an MVP definition into user stories that act as the common language between all participants in the development process. A user story envisions and describes a feature in which user value is delivered incrementally. Stories also include acceptance criteria, which are requirements needed to complete the story.

WHO CAN DO WHAT AND HOW?

Who is the user that benefits? In the Enterprise Design Thinking workshop, you identified your users. Write your stories using the name of the persona and not their role. This is straightforward, direct, and makes it easy to extract meaning from the backlog.

What can the user do that they could not do before? Maybe the user can see some important information or take an action that helps them to achieve something. Stating exactly what the persona can do leads to stories that read more naturally. This allows you to better empathize with the user and makes it easier to write good stories.

How does the change benefit the user? Sometimes the benefit is obvious. If it isn't, state it explicitly.

Learn more

Read about writing user stories.
https://www.ibm.com/garage/method/practices/think/user-stories

If a perfect story existed - this would be it!

User stories - the good, the bad, and the ugly

The easiest way to learn to write user stories is to review examples of poorly written stories and learn how they can be improved.

PRACTICE, PRACTICE, PRACTICE

The not-great story has a poor signal to noise ratio. The actual information is hidden behind the “As a” construct that has no meaning and the “So that” construct is redundant. In the better version, the Persona Mike is called out directly. The “can” phrase is more story-like and describes what Mike needs and why.

Not so good

As a mechanic, I want to see a list of parts suitable for my vehicle so that I don't have to look at irrelevant parts.

Better

Mike, the mechanic, can see a list of parts suitable for his vehicle.

Learn more

Read about writing user stories.
https://www.ibm.com/garage/method/practices/think/user-stories

In the bad story, the boilerplate might look like a user story but it is not. The user should not be the person who has to implement the story. The feature described in the story should be a delightful capability, not a burdensome obligation. In the better story, Mary benefits from the feature for the reason stated in the story.

Bad

As a developer, I need to write documentation.

Better

Mary, the integration developer, can create new flows by referencing documentation.

In the bad story, a developer would have no idea how or what to implement. In the better story, there is a concise description of the user, Pat, the function he needs, and why he needs it.

Bad

Make a dashboard.

Better

Pat, the platform specialist, can track and monitor stats of the plan to check that a flow is performing its function.

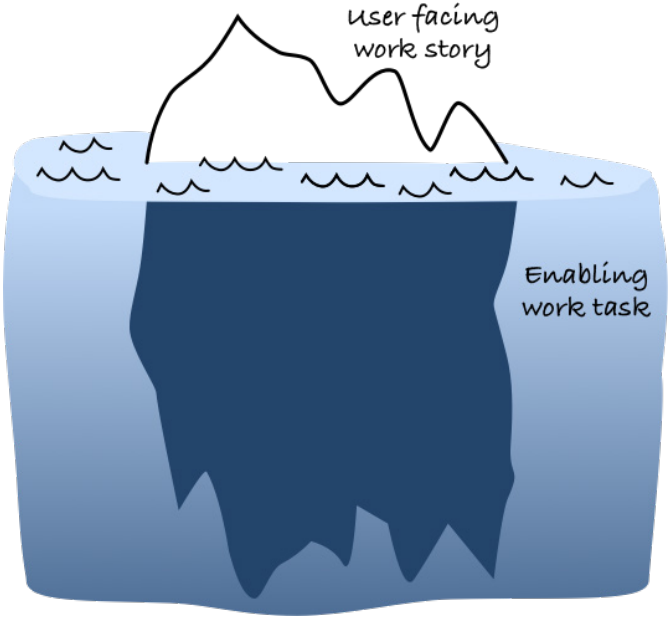
Sometimes a user feature is driven when something happens in the system vs a specific user action. This story reflects that when an order event is received by the system, Frank is notified with a sparkler on his dashboard. The story does not specify the design for what is shown, just that there will be a visual indicator of the event.

Job story

When an order is received, Frank sees a "sparkler" on his dashboard.

Tasks, non-functional requirements and bugs

Sometimes a squad does work related to the application they are developing and the result does not impact the application user. This work should not be written as a user story. Capture it as a task (aka a chore) in the backlog. The squad lead and the product owner prioritize the backlog to ensure a balance of user stories and tasks.



Learn more

Read about writing user stories.
https://www.ibm.com/garage/method/practices/think/user-stories

TAKE CARE OF THE PLUMBING!

Use tasks to take care of the plumbing. Tasks improve code quality, maintainability, or development efficiency. Tasks are used to break down iceberg stories into smaller pieces. Plumbing tasks might include acquiring and configuring a cloud environment for the application testing and production environments, integrating analytics to track site usage, etc.

Track non-functional requirements in your backlog. The development team is responsible for writing, prioritizing, and accepting tasks when the task supports a business goal or non-functional requirement. Non-functional requirements (NFRs) include automating the delivery process to implement DevOps practices and tools, security, build-to-manage functionality, and resilience for backup, recovery and disaster recovery.

Fix your bugs. Problems that are found after a story is delivered, including user related problems, performance, or other NFR isssues are bugs. Capture your bugs as tasks and prioritize them in the backlog. If a bug is preventing the delivery of a user story that is under development, it should be prioritized high so that the user story can become whole and be delivered.

The definition of done

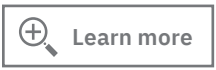
What does ‘done’ mean? In practical terms, if you can’t forget about a task when it reaches the “Done” column of your board, it is not done. Acceptance criteria defines the set of requirements that must be met for a story to be marked complete. The definition of ‘done’ is a set of common criteria that must be met to close all stories.

HOW DO I KNOW WHEN I’M DONE?

Include design. Hold a design review to make sure the implementation matches the design. Depending on the project priorities, it might not be necessary to be pixel perfect but the team should discuss expectations.

Apply standards to all stories. The team defines standards that all stories must meet. Examples include design review, monitoring, responsiveness, and security standards. Done criteria can also include acceptance testing, storage of code in source control, feature testing, and deployment to production.

Think bigger – measure the value. Every user story must deliver value to be considered done. Gather metrics to measure the value being delivered to users. Talk to your stakeholders to ensure you have their buy-in for the function you deliver.



Read about the definition of done.
https://www.ibm.com/garage/method/practices/culture/definition-of-done

The journey to “done” can take a number of paths.

Manage stories using Kanban

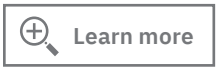
After you've defined your MVP and broken down your stories through an Inception workshop, you must manage the work to build out the MVP and ensure that everyone on the team understands what needs to be done. Use a Kanban board to keep everyone on the same page.

NEW, BACKLOG, IN-PROGRESS, AWAITING ACCEPTANCE, DONE

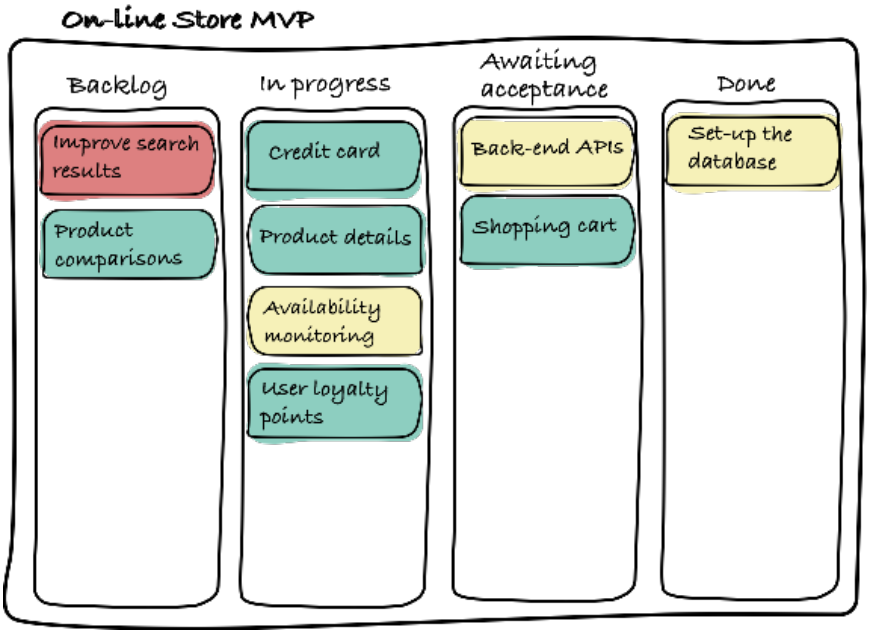
Define your stages and create Kanban columns. The squad decides on the columns on your Kanban board. It is critical to how the squad will manage their work. Ensure acceptance and done criteria are included in how the team works with the board. A story cannot move to complete until acceptance and done criteria are met.

Manage a rank-ordered backlog. Rank user stories that are waiting to be started in priority order. Developers pull user stories from the top of the backlog. Adjust story priorities in the backlog based on feedback and new requirements.

Frequently groom your backlog. Capture new stories in your tracking system so they don't get lost. Incorporate them into the backlog as part of your prioritization discussions.



Learn how to track stories on a Kanban board.
https://www.ibm.com/garage/method/practices/culture/practice-kanban-method/



Progress your stories through the Kanban board as your squad works to complete your MVP.

Iterations in Garage Methodology versus Scrum

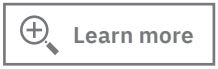
The methodology combines best practices from a variety of approaches to create a new way of working. Some practices are similar to those in the Scrum framework; however, there are key differences. The practices improve delivery efficiency and assure that communication through planning, playback, daily standups, and retrospective analysis occur on a regular cadence.

PLAN AND PLAYBACK REGULARLY. DELIVER CONTINUOUSLY.

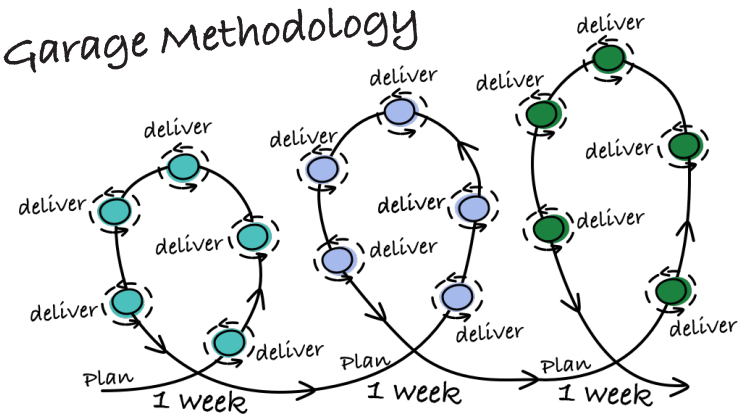
Deliver continuously. In the Scrum framework work is planned, assigned, and delivered within each sprint boundary. In the methodology , the focus is on continuous delivery. As a user story or task is completed, it is delivered.

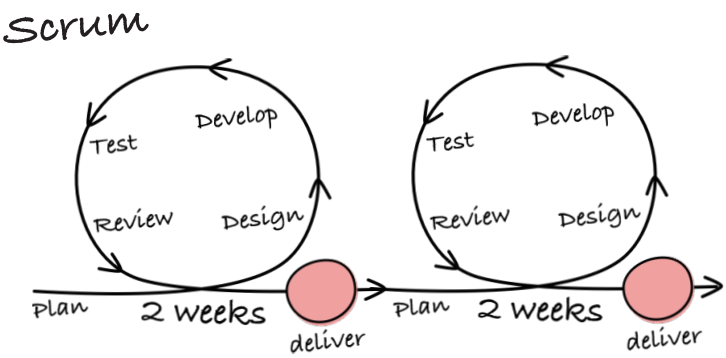
Assign stories and tasks just in time. In Scrum, assignments are made on the sprint boundaries. In the methodology, after a user story or task is delivered, another is selected from the top of the prioritized backlog, assigned to the team member who is available, and the work is started immediately.

Iteratively plan and playback. In Scrum, planning and playbacks occur only on sprint boundaries. In the methodology, there is always an iteration in progress. An iteration describes one week of development work, but there are no hard boundaries within which coding tasks have to fit. Planning, playback, and retrospectives occur on a weekly basis.



Learn about iteration planning in the IBM Garage Methodology.
https://www.ibm.com/garage/method/practices/code/practice_iteration_planning/





With the method, there is less ceremony and planning at the iteration boundaries.

Iteration ceremonies

Iteration ceremonies are routine events that keep iterations organized and help to ensure that the right work is completed. Four common ceremonies are iteration planning, daily standups, retrospectives, and playbacks. Ideally, the team rotates who leads the ceremonies. For people unfamiliar with leading, start them with the daily standup.

LIMIT MEETINGS, NOT COMMUNICATION

Plan your work for the week. Review the stories in the backlog. Prioritize new work that has been added and determine the stories the team will work on in the coming week.

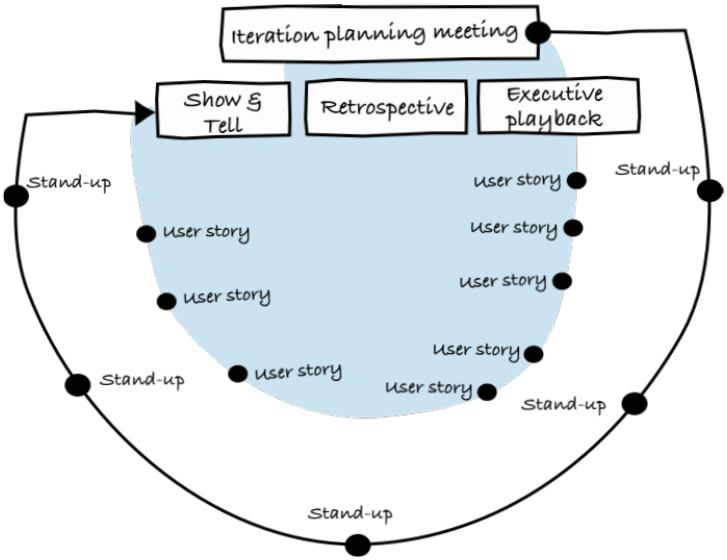
Keep the team in sync with daily standups. Review the day's progress, report on what's next, assign pairs, and identify blockers that are impeding progress.

Playback completed work to get feedback. Conduct meetings within the team or with project stakeholders to showcase completed work. Capture feedback to make improvements.

Hold retrospectives. Reflect on what went well and what did not. Capture improvements as tasks in the backlog.

Add meetings only when necessary. Hold design and architecture reviews only when needed. If possible, see if reviews can fit into one of the regularly scheduled meetings.

Learn about agile ceremonies.
https://www.ibm.com/garage/method/practices/culture/agile-ceremonies



Planning and reflection meetings bookend the week. User stories are delivered on a steady cadence.

Iteration planning meeting

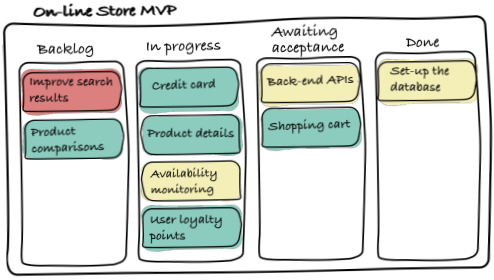
During iteration planning, the product owner and the team review the ranked backlog and decide what to accomplish in the coming week. At the end of a successful planning meeting, the team knows the stories that are next in the backlog and they have had an opportunity to ask their questions.

IDENTIFY, PRIORITIZE, CLARIFY, AND ASSIGN

Identify and write missing stories. Before the planning meeting, the product owner grooms the backlog by identifying and writing missing user stories and ranks the backlog. This includes taking into account any new stories or tasks.

Review the priority. The product owner and the squad review the relative priority of the items in the backlog and adjust as needed.

Come to a shared understanding of what's coming next. In the planning meeting, the team reviews the upcoming stories. Questions are answered and clarifications are made to ensure everyone leaves with a common understanding of the work that needs to be done.



Learn more

Conduct an iteration planning meeting.
https://www.ibm.com/garage/method/practices/code/practice_iteration_planning

Daily stand-up meeting

A successful daily stand-up ensures that there is always open and honest communication across your team. The discussion focuses on 3 questions including: what have you accomplished since the last stand-up, what are your goals the day, and what is blocking your progress. If your team pair programs, discuss who pairs with who.

3 QUESTIONS, 3 ANSWERS, NO MORE THAN 15 MINUTES

Align with assigned stories and tasks. Ensure accomplishments are in line with assigned work. If work spans multiple days, be sure to provide details that show progress towards completing the work.

Keep laser focused on time. The goal is to keep your daily stand-up meetings concise and no longer than 15 minutes.

Do not resolve blockers in the stand-up. Schedule additional time at the end of the meeting or hold a separate meeting to resolve blockers. The key is that any person who is not involved in the issue can leave the meeting and return to their work.

Pair rotation. If your team is using pair programming, discuss upcoming pair rotation assignments in the standup meeting.

1. What did you accomplish yesterday? "Yesterday, we wrote all the test cases and got half way through the implementation."

2. What do you aim to accomplish today? "Today, we will finish the implementation and start the credit card story."

3. Is there anything preventing your progress or slowing you down? "This work has no blockers."

Retrospectives

Retrospectives, or retros, provide a chance to uncover what is working well for a team and what isn't working well. Teams have a chance to step back from day-to-day work and reflect on the big picture. During a good retro, people feel safe sharing their opinions so that the team can improve the way they work.

CHECK THE TEAM'S PULSE

Collect input from the team. Using a poster with sticky notes or a MURAL board, the team takes about 20 minutes to write items that are working well, items that are causing problems, and items that need discussion. After input is collected, the moderator facilitates discussion so the team understands the impact of the input.

Cluster and vote. The moderator clusters the items to eliminate duplicates. The team is then given 3 – 5 votes used to identify the items that are most important to them.

Discuss and take action. After voting, the team discusses the items with the most votes to identify actions that can be tracked in the backlog. The key to success is completing the work that was identified as problematic and improving how the team works.

Learn more

Conduct a retro.
https://www.ibm.com/garage/method/practices/learn/practice_retrospective_analysis

Playbacks

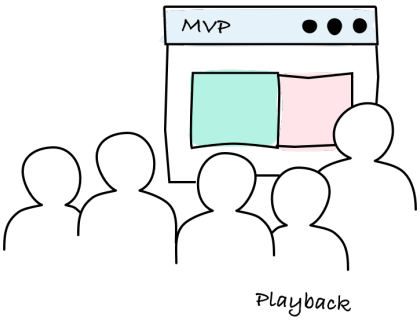
Playbacks are presentations that align your squad, stakeholders, and clients on a user experience. Playbacks take two forms. Internal playbacks align the squad to make sure they are moving in the right direction. External playbacks align the squad and stakeholders throughout the project. In this meeting, the product owner might share concise demonstrations of the running code, clickable prototypes, or wireframes of the minimum viable product (MVP) to get feedback from stakeholders and sponsor users.

SHOW OFF YOUR WORK

Match the playback style to the audience. What you play back informally to your squad might differ greatly from the format that you play back to executive stakeholders. Make sure the playback presentation is tailored to your audience.

Solicit and gather feedback throughout the playback. Schedule enough time to allow playback attendees to provide feedback throughout the presentation. Capture the feedback for review by the team.

Incorporate playback feedback into your backlog. Discuss the feedback from the playback. Open additional user stories and tasks so that you can improve what you are doing based on direct input from your stakeholders.



Building effective squads

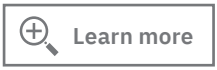
A squad is a small independent team with a squad leader, who acts as the anchor developer and agile coach, and 3 or 4 development pairs. The squad works together to accomplish a common goal, which is to build their assigned part of the MVP.

ALL FOR ONE AND ONE FOR ALL

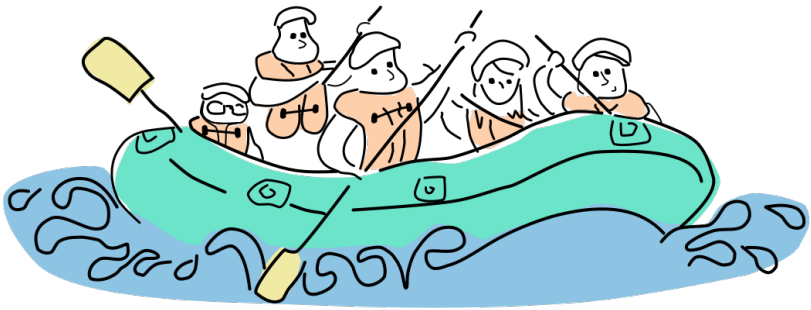
Communicate and work towards a common goal. Successful squads communicate well. Through communication, everyone on the squad gains an understanding of the squad's common goal and stays in sync.

Develop and deliver user stories. Squad members work together to implement user stories based on their priority in the ranked ordered backlog.

Pair program. Squads benefit from using pair programming. When pairs write code together, it undergoes continuous code review, making it possible to reduce or eliminate formal code reviews. Rotating programming pairs daily spreads the knowledge of individual system elements across the squad. Code is continually read, revisited, and revised as new user stories are implemented. Pair programming reduces the dependence on any single person on the squad. Using pair rotation with test-driven development makes it possible for any squad member to participate in a pair with confidence.



Build a high performance squad.
https://www.ibm.com/garage/method/practices/culture/practice-building-effective-squads

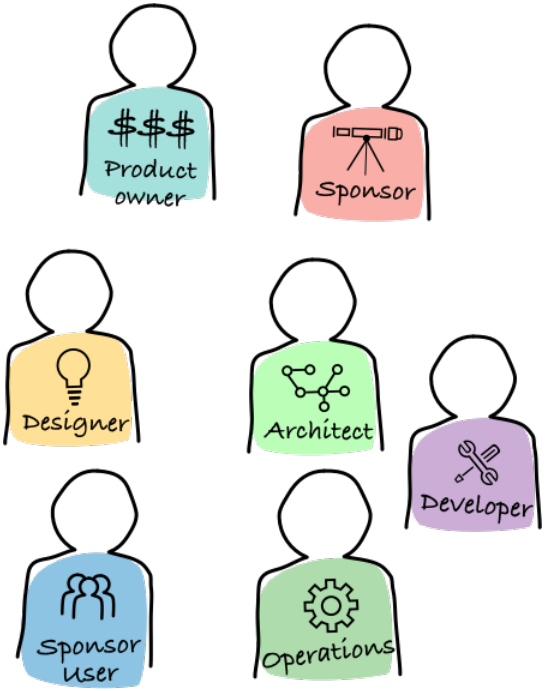


Depending on the needs of the project, a squad may include other specialists such as data scientist, and post-launch support specialists.

Define organizational roles

There are specific organizational roles for a squad to be effective.

WHO ARE THE PLAYERS?



Learn more

Learn about the roles you need for success.
https://www.ibm.com/garage/method/practices/discover/practice_organizational_roles

Product owner. Owns the product vision. Writes user stories and prioritizes the backlog. Makes product strategy decisions. Engages the team to create a product that delivers business value by meeting the needs of the market. Looks at analytics to ensure the team is meeting goals.

Sponsor. Ensure that the squad has everything it needs to succeed. This is usually an executive with a vision who owns the overall delivery and success of the project.

Architect. Investigates key technologies and approaches. Considers how to achieve non-functional requirements (NFRs). Creates the architectural designs, diagrams, and documents that are actively used by the squads to guide their development. Envisions the evolution of the solution.

Designers. The goal of every project is to create experiences that delight users. The UX design lead is responsible for all aspects of the project's user experience. The visual designer converts the user experience concepts into detailed designs that emotionally connect with users.

Developers. Build code by using core agile practices such as "keep it simple," test-driven development (TDD), continuous integration, and pair programming. Developers must be adept at rapidly learning and using new technologies.

Service management and operations roles. Implement processes to manage operational aspects of the application. Members include first responders, site reliability engineers, problem analysts and subject matter experts.

Sponsor users. A sponsor user is a real user—someone from outside your organization—who will use the product. Select your sponsor users carefully so they accurately represent the needs of the widest possible user base.

Roles in a squad at scale

The structure of your squads depends on the size and complexity of the MVP under development. Squads are responsible for building the deployment platform, developing application functionality, ensuring that the entire solution is reliable.

KNOW WHO DOES WHAT TO WHOM

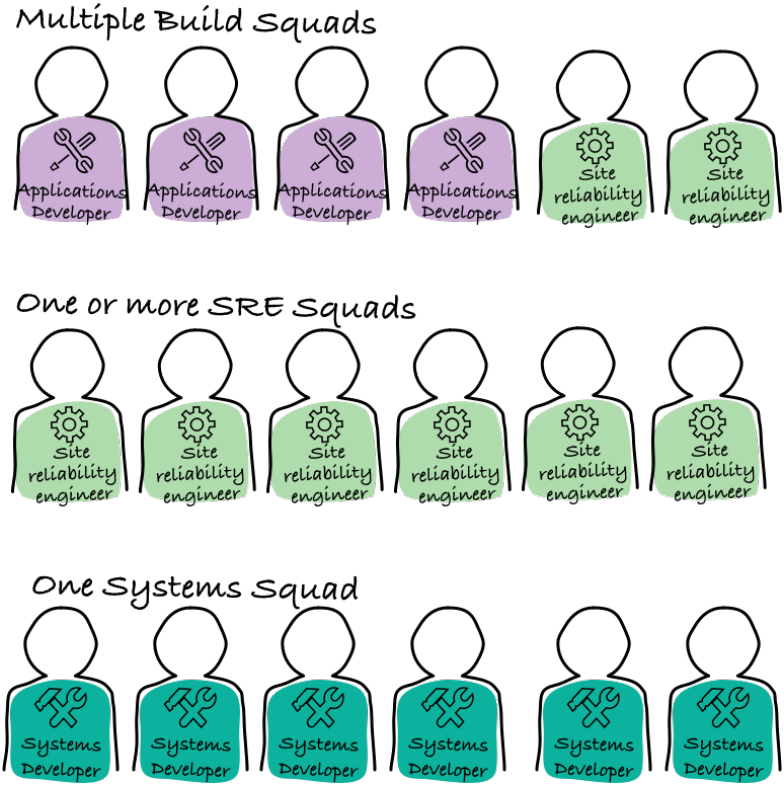
Design and develop the application. Staff your squads with skilled application designers and developers who are familiar with the languages and technologies you are using.

Ensure the application works. Site reliability engineers who are embedded in the development (build) squad build out the operational aspects needed to manage your application in production. They are responsible for setting up monitoring and logging configuration, building runbooks, and application performance and reliability testing.

Ensure the overall system works. System reliability engineers address cross-cutting concerns that affect the entire system. Their responsibilities include reliability and performance testing across the whole system. Additionally, they setup, maintain, and manage the tools needed for development and operations.

Deploy, configure, and run the platform. Systems developers choose, purchase, install, and configure the application infrastructure, including the shared automation infrastructure.

Understand the interaction between development and operations.
https://www.ibm.com/garage/method/practices/culture/practice-building-effective-squads/roles-in-a-squad



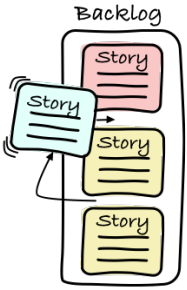
As you scale your project, you also scale your squads.

Lifecycle of a user story

Every user story envisions and describes a feature that delivers user value. The product owner initiates each user story and tracks it throughout its lifecycle. Stories are written on an ongoing basis, sometimes as the result of feedback from your sponsor users and stakeholders. It is important to continuously groom your backlog by inserting new stories and reprioritizing existing stories as you learn throughout the project.

USER STORIES HAVE A LIFE OF THEIR OWN.

Product Owner writes a new user story with guidance from the delivery team.



Product Owner prioritizes the new story, moving it into the Backlog so the delivery team knows where it fits in their work.

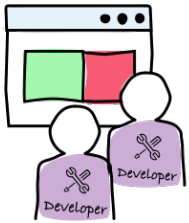
Learn more

Visit the Garage.
https://www.ibm.com/garage

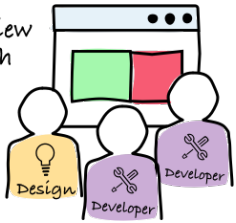
As the new story moves to the middle of the backlog ...

... the designer produces visual designs and attaches them to the story.

Developers implement the story.

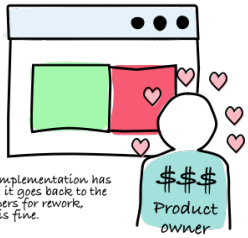


Developers review their work with the designer.



Now implemented, the story waits for approval with instructions for trying it out

Product Owner reviews the implemented story and, if satisfied, they approve it.



If the implementation has issues, it goes back to the developers for rework, which is fine.

Working together on a project

Co-creation is an important part of the IBM Garage Methodology; co-creation means everyone in a squad collaborates on a common goal and communicates often. Development and design work at the story level. The squad's architect will work at a level above stories. Design reviews are an important mechanism for collaboration on a project.

DEVELOPMENT AND DESIGN ARE JOINED AT THE HIP

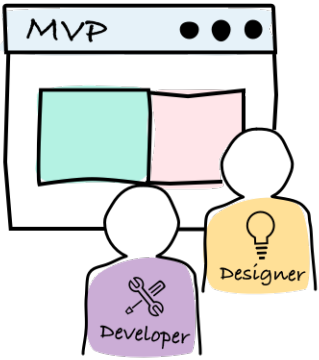
Development and design collaboration. Development-design collaboration works best when it's fluent and regular. It can work well for designers to sometimes pair with developers to fine tune the UI.

Vision reviews. Conduct meetings with the product owner to review designs for brand fit and vision. Discuss the results of user research.

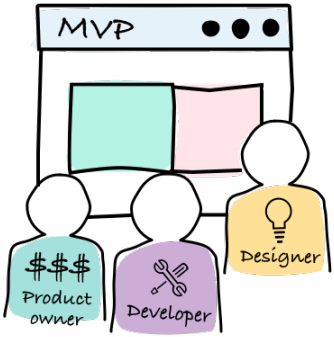
Implementability reviews. Development reviews the technical difficulty of the design before the design is finalized. Development can provide input on alternatives to costly features.

Implementation reviews. Designers should review what's been implemented to spot differences between what's been implemented and the original design. Depending on the hypothesis being tested, developing a pixel-perfect UI might not be the wisest use of a squad's time - but the finished product shouldn't be a complete surprise to the designer.

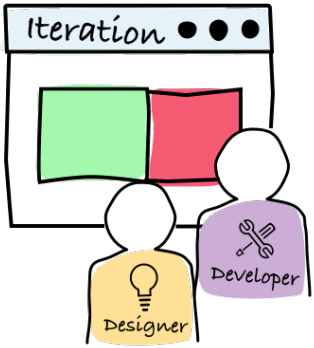
Check out co-create in the Garage Methodology.
https://www.ibm.com/garage/method/co-create



Design collaboration



Vision review



Implementation review

To co-create effectively, be sure to include the three types of design review.

Fun at work

It's easy to get so caught up in the day-to-day grind of delivering stories, that the team can lose sight of a key to productivity. FUN! Although everyone likes fun, it's often seen as unprofessional - fun is something that happens outside work. But, fun matters. The evidence that fun boosts productivity is compelling. A happy work environment means fewer sick days, harder work, and greater productivity, up to 31% greater for team members with a positive mindset.

A FEW LAUGHS MAKE FOR EFFECTIVE EMPLOYEES

Remove the unfun. Identify the daily grind of wasteful activities that drain morale. As you use the method to focus on increasing feedback and removing cost, remove many of the tedious low-return activities that take developers away from delivering customer value.

Move on to having fun. Incorporate fun into work. Share adventures at the end of your playback meetings. Take a break and walk, or play a game with a co-worker. Build fun into your processes through gamification, exploration, and creativity.

Having fun when remote. While you cannot walk or play ping-pong with someone when working remote, you can still incorporate fun into your days. Make time to chat about your day with a friend at work. Schedule zoom breaks or create a discussion group where your team can just meet and chat.

Don't forget to have fun!
https://www.ibm.com/garage/method/practices/culture/fun-in-the-workplace

Celebrating success is motivating for your team - and it’s fun!

Visit the IBM Garage to learn how IBM can help you achieve your business goals.

https://ibm.com/garage

Learn about the IBM Garage Methodology.

https://www.ibm.com/garage/method

Notices

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

IBM Garage Day-to-Day