What is software development?

Software development refers to a set of computer science activities dedicated to the process of creating, designing, deploying and supporting software.

Software itself is the set of instructions or programs that tell a computer what to do. It is independent of hardware and makes computers programmable. There are three basic types:

System software to provide core functions such as operating systems, disk management, utilities, hardware management and other operational necessities.

Programming software to give programmers tools such as text editors, compilers, linkers, debuggers and other tools to create code.

Application software (applications or apps) to help users perform tasks. Office productivity suites, data management software, media players and security programs are examples. Applications also refers to web and mobile applications like those used to shop on Amazon.com, socialize with Facebook or post pictures to Instagram.

A possible fourth type is embedded software. Embedded systems software is used to control machines and devices not typically considered computers — telecommunications networks, cars, industrial robots and more. These devices, and their software, can be connected as part of the Internet of Things (IoT).

Software development is primarily conducted by programmers, software engineers and software developers. These roles interact and overlap, and the dynamics between them vary greatly across development departments and communities.

Programmers, or coders, write source code to program computers for specific tasks like merging databases, processing online orders, routing communications, conducting searches or displaying text and graphics. Programmers typically interpret instructions from software developers and engineers and use programming languages like C++ or Java to carry them out.

Software engineers apply engineering principles to build software and systems to solve problems. They use modelling language and other tools to devise solutions that can often be applied to problems in a general way, as opposed to merely solving for a specific instance or client. Software engineering solutions adhere to the scientific method and must work in the real world, as with bridges or elevators. Their responsibility has grown as products have become increasingly more intelligent with the addition of microprocessors, sensors and software. Not only are more products relying on software for market differentiation, but their software development must be coordinated with the product's mechanical and electrical development work.

Software developers have a less formal role than engineers and can be closely involved with specific project areas — including writing code. At the same time, they drive the overall software development lifecycle — including working across functional teams to transform requirements into features, managing development teams and processes, and conducting software testing and maintenance.

The work of software development isn't confined to coders or development teams. Professionals such as scientists, device fabricators and hardware makers also create software code even though they are not primarily software developers. Nor is it confined to traditional information technology industries such as software or semiconductor businesses. In fact, according to the Brookings Institute (link resides outside of ibm.com), those businesses "account for less than half of the companies performing software development."

An important distinction is custom software development as opposed to commercial software development. Custom software development is the process of designing, creating, deploying and maintaining software for a specific set of users, functions or organizations. In contrast, commercial off-the-shelf software (COTS) is designed for a broad set of requirements, allowing it to be packaged and commercially marketed and distributed.

Steps in the software development process

Developing software typically involves the following steps:

Selecting a methodology to establish a framework in which the steps of software development are applied. It describes an overall work process or roadmap for the project. Methodologies can include Agile development, DevOps, Rapid Application Development (RAD), Scaled Agile Framework (SAFe), Waterfall and others. (See the glossary.)

Gathering requirements to understand and document what is required by users and other stakeholders.

Choosing or building an architecture as the underlying structure within which the software will operate.

Developing a design around solutions to the problems presented by requirements, often involving process models and storyboards.

Building a model with a modelling tool that uses a modelling language like SysML or UML to conduct early validation, prototyping and simulation of the design.

Constructing code in the appropriate programming language. Involves peer and team review to eliminate problems early and produce quality software faster.

Testing with pre-planned scenarios as part of software design and coding — and conducting performance testing to simulate load testing on the application.

Managing configuration and defects to understand all the software artifacts (requirements, design, code, test) and build distinct versions of the software. Establish quality assurance priorities and release criteria to address and track defects.

Deploying the software for use and responding to and resolving user problems.

Migrating data to the new or updated software from existing applications or data sources if necessary.

Managing and measuring the project to maintain quality and delivery over the application lifecycle, and to evaluate the development process with models such as the Capability Maturity Model (CMM).

The steps of the software development process fit into application lifecycle management (ALM). The IBM Engineering Management solution is a superset of ALM that enables the management of parallel mechanical, electrical and software development.

Requirements analysis and specification

Design and development

Testing

Deployment

Maintenance and support

Software development process steps can be grouped into the phases of the lifecycle, but the importance of the lifecycle is that it recycles to enable continuous improvement. For example, user issues that surface in the maintenance and support phase can become requirements at the beginning of the next cycle.

KT2502 Game development

Game development, like any other project, starts with an idea. The hard part is to materialize this idea into a shippable product. To achieve this, we follow specific steps.

1. DISCOVERY

It is the very first step where the idea turns into a game. Initially, the discovery stage identifies a game's scope and what's required to bring it to a release date.

The discovery stage usually lasts for a month or two, depending on the project scope, resources needed, and available budget. At this point, the team consists of a few members – the Core Team. The Core Team usually consists of a Solution Architect, Game Designer, and Art Director.

The discovery stage is a vital starting point of the game creation and usually implies a wide range of activities displayed in detail below:

Game Design Document Creation. Designers create a single document that has all the rules and descriptions of the game, namely:

Game mechanics description

User flow UX charts

Customization design

Metagame design

Art-Style Definition. In collaboration with Game Designers and Stakeholders, Art Director, and the teamwork on the visual style for the product, developing the aesthetics, look, and feel.

Game visual style

Concepts (background/environment, characters, some iconography)

3D models (characters, maybe some props like weapons)

3D animations (if we have time)

VFX concepts/prototypes (how particle effects might look like)

Requirements for graphics

Polygon budget (based on target devices)

Texture resolution

Animation requirements (animation skeleton requirements like number of bones, naming convention, folder structure)

Technology Stack Definition. At this point, Solution Architect or Tech Lead identifies the technologies needed for the development process.

Define if the team can use an existing solution

Usually, it's faster to develop a product with an available game engine

In some instances, it makes sense to build a custom engine or platform

Pick among various available engines

Decide what engine suits best to express your idea

Define the project architecture

Make sure the product is built with expansion and future support in mind

Planning. When the team is ready with the procedures mentioned above, we provide the information on:

The team needed to develop the game

How long it takes to develop a game

Deliverables produced along the way

Budget breakdown

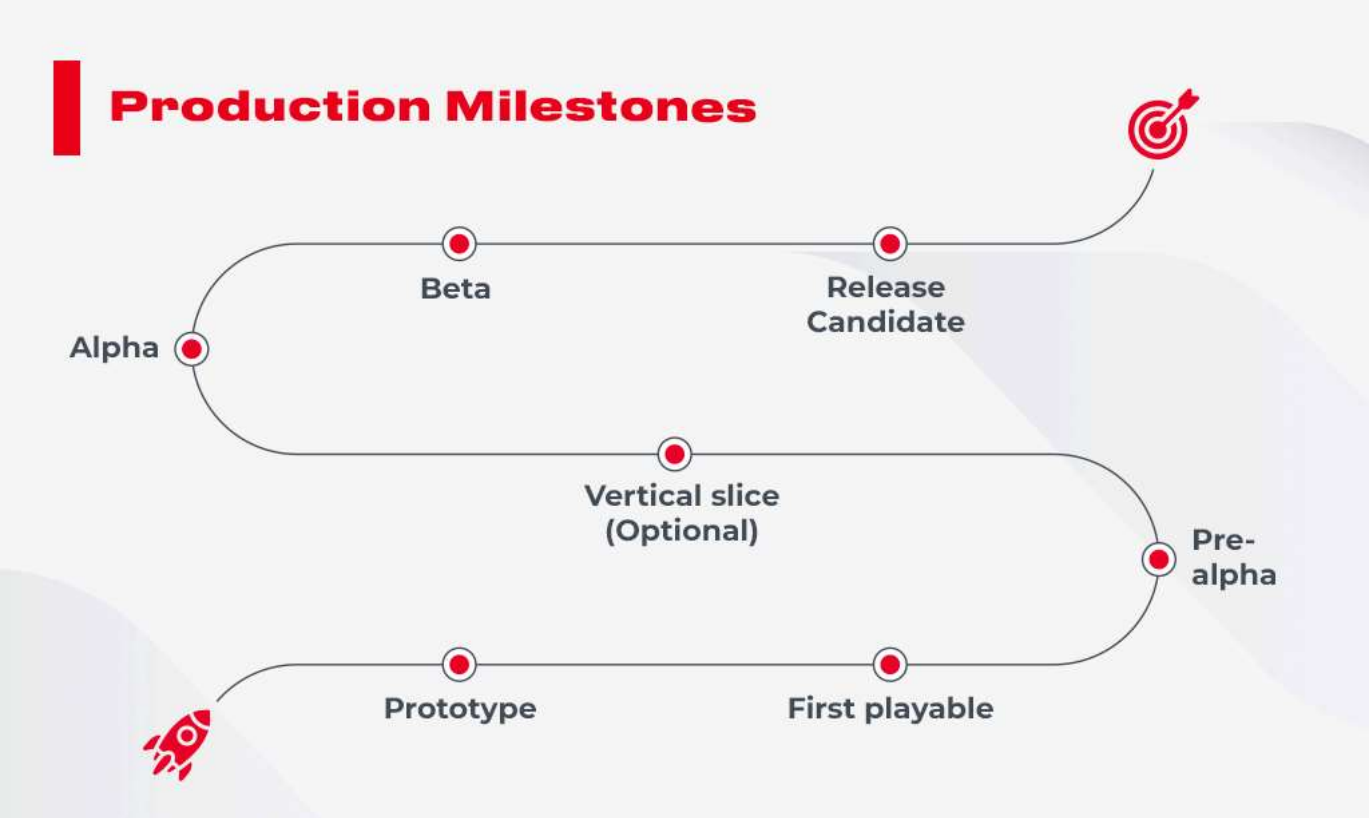
2. PRODUCTION: THE MOST LABOR-INTENSIVE STAGE OF THE GAME DEVELOPMENT PROCESS

This is when the actual development begins. Team members polish the story, define game mechanics, its balance, pacing, and gameplay. Plus, they create all assets (characters, creatures, props, and environments), set the rules of play, build levels and scenes, and write the code.

Each component of the game has to be designed thoughtfully, the fun and gameplay, and then characters, environment, objects, the level of difficulty, scenes, and more. Because the initial ideas don't always render well in reality, the game testing and improvement continue even when the game is released.

Let's look at the fundamentals of game production, and some of the critical video game development jobs, bearing in mind that small teams undertake multiple roles. In contrast, a larger studio with more team players who specialize in a particular aspect of production.

Production Milestones



Prototype: A video game prototype is a raw test that examines functionality, user experience, gameplay, mechanics, and art style. Prototyping happens as the first phase of production to test whether the game idea will work and is worth pursuing. Many ideas fail at this stage.

First playable: The first playable allows us to get a better idea of the game's look. While it is still far from the final version, placeholders are starting to be replaced with higher quality elements, and art is added.

Pre-alpha: Most of the content is designed in the pre-alpha stage. At this point in the game development process, some critical decisions take place. The content may get cut, or new assets are incorporated to improve gameplay.

Vertical slice (optional): A vertical slice is a fully playable version that can be used to manifest your game to clients, studios, or investors. A vertical slice gives a first-hand experience of the game, ranging from a couple of minutes to half an hour.

Alpha: The game is packed with all features, meaning that it is entirely playable from start to end. Some elements, such as art details, may still need to be added, but controls and functions should be working accurately.

Beta: Here, all the content and assets are aligned, and the team should focus on improvement rather than adding new functions or features. The QA testers ensure everything is running seamlessly and report bugs back to the side.

Release Candidate: The game is ready-to-use and can be sent to the publishing outlet and launched to the public.

The 10 Key Roles In The Game Development Process

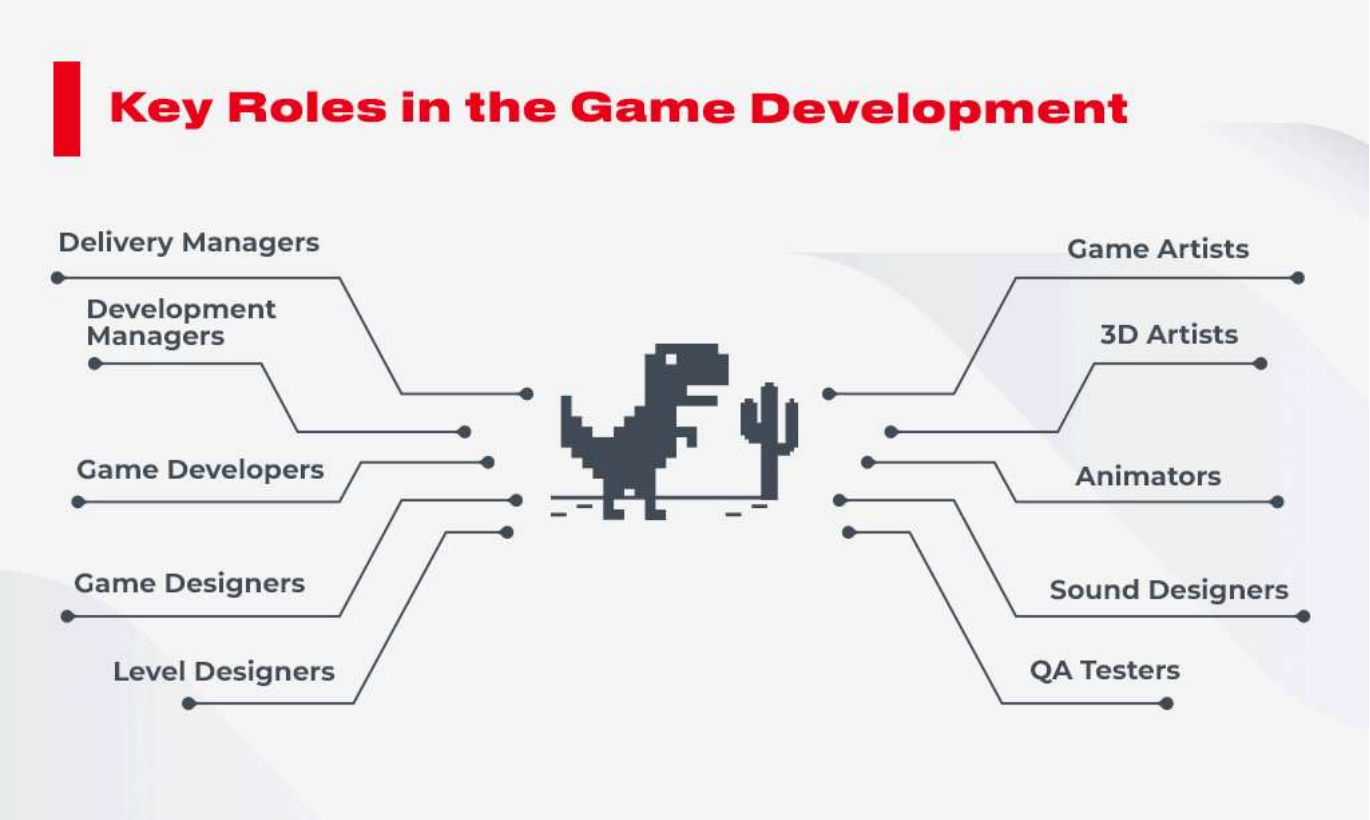
Delivery Managers / Producers manage the business aspect of the project, particularly the expenses and schedules. Producers typically handle the budget and develop marketing strategies to sell the product.

Development Managers ensure game development runs smoothly, milestones are aligned, risks are mitigated, and teammates are doing what they're supposed to. A project manager is often in the center of communication between the dev and design teams and executives.

Game Developers build games by turning design concepts into fully playable games. Game developers have a solid programming background. Plus, devs should have a combination of creativity, math skills, and patience to code ideas into interactive visuals and sounds.

Game Designers are the creative drivers of the game, and bridge between writers and artists, with some knowledge of programming. The game design production includes creating attention-grabbing stories, characters, goals, rules, and challenges that drive interactions between game elements.

Level Designers are responsible for creating exciting and fun levels. The challenge a level designer has to cope with is to keep a player's focus and achieve the goal while reducing the potential of confusion. Level designers are also responsible for identifying player's bottlenecks when going through the game, such as falling out of borders or getting stuck and not being able to get out.



Game Artists encompass concept artists, animators, 3D modelers, and FX artists. They usually create concept art, 2D elements, give color, action, and life to the play.

While concept artists are active during designing of the first look, they may be involved later in the game development process to add new elements into the game.

3D Artists create models of characters, objects, props, and environments that can then be textured and animated. Modelers need to know how to get and use high-quality reference materials, primarily if they're replicating real objects like, for instance, Kalashnikov rifle.

Animators make characters and objects lifelike by adding movement to them. They create storyboards and outline key animation scenes that match the game's storyline. Animators often conduct lots of research on the objects they need to bring to life.

Audio Engineers / Sound Designers make natural sound effects, record voice dialogues between characters, create soundtracks to set the mood for players, i.e., opening music, menu pause music, marking a victory.

QA Testers are vital in the game development process. These people check games for bugs and make sure the game runs seamlessly, and guides are clear for players. They report errors to the developers in what's sometimes known as a bug sheet.

3. TESTING

Every feature and mechanic in the game requires testing for quality control. A game without thorough testing is a game that's not even ready for an Alpha launch. Here are some questions a QA tester may check during this stage:

Are there error-areas or levels?

Is everything rendering on the screen?

Can my character walk through a particular wall?

Does my character get forever stuck at this point?

Is the character dialogue boring?

There are even various types of playtesters. Some of them conduct stress tests by running into walls hundreds of times to crash the game. Other play-testers check "fun factor" to see if the game is too complicated or too easy and comfortable enough.

After countless iterations of testing, the game should be ready for Alpha or Beta release, depending on how refined the in-game features are. At this phase, the players, for the first time, get hands-on the game.

4. RELEASE

The release day is coming, and one can see the light is at the end of the tunnel. The months leading up to an awaited date is typically spent in errors debugging found in the testing stage.

In addition to bug squashing, developers hone in on the game as much as possible before it launches. Maybe that rock can have more depth. Perhaps the character's outfit can be more texture, or those trees may finally sway in the wind. These slight changes can be significant for making a video game more immersive.

KT2503 Software tools

Most useful software development tools

A software development tool is a program that can be leveraged to develop, maintain, or test other applications. Today there are both paid and free software development tools available to choose from. There are a wide range of software development tools, and each of them perform a specific role better than the others.

Software development tools are in huge demand. The software industry has been experiencing phenomenal growth in recent years. In 2020, the IT industry was worth $ 5.2 trillion and it is obvious why so many people want to be a part of such a rewarding industry. Even with over 21 million developers out there, the industry still faces a shortage due to a lack of skilled professionals. Here we have discussed the most useful software development tools that are available today.

1. Github
This is the most popular one out there. It is a web-based Git repository hosting solution that serves as a place where developers can search for codes. The platform allows developers to upload private/public projects and helps to keep them safe. Github has a large community and an impressive repository of projects. It is the perfect solution for professionals who seek to collaborate.

2. Gleek
It is similar to Github as it is a data modeling tool and not just a typical software tool. It is free to use and can be leveraged without signing up for an account. It does not provide any drag and drop functionality and it can be accessed using a keyboard.

3. Codepen
It is a powerful tool especially helpful for front-end developers. The tool is a great place to find ideas for your next project. It allows web developers to share CSS, HTML related issues to find a quick solution from other developers.

4. Buddy
Another web development tool that leverages delivery pipelines to develop, deploy and test software. It is easy to use due to the action system that allows developers to arrange it in the required way. It is perfect for deployment-related tasks. It takes as low as only 15minutes to configure. The tool works flawlessly with WordPress, Google, AWS, etc.

5. Cloud9 IDE
It is an online integrated environment where you can find answers to your app development-related problems. The tool offers support for a vast range of languages such as Python, Ruby, PHP, C, and JavaScript. It can be leveraged to set breakpoints as it saves time and it is quite user-friendly. Tool's code completion section offers useful suggestions which help in writing codes faster.

KT2504 Mobile Apps

What Does Mobile Application (Mobile App) Mean?

A mobile application, most commonly referred to as an app, is a type of application software designed to run on a mobile device, such as a smartphone or tablet computer. Mobile applications frequently serve to provide users with similar services to those accessed on PCs. Apps are generally small, individual software units with limited function. This use of app software was originally popularized by Apple Inc. and its App Store, which offers thousands of applications for the iPhone, iPad and iPod Touch.

A mobile application also may be known as an app, web app, online app, iPhone app or smartphone app.

Mobile Application (Mobile App)

Mobile applications are a move away from the integrated software systems generally found on PCs. Instead, each app provides limited and isolated functionality such as a game, calculator or mobile web browsing. Although applications may have avoided multitasking because of the limited hardware resources of the early mobile devices, their specificity is now part of their desirability because they allow consumers to hand-pick what their devices are able to do.

The simplest mobile apps take PC-based applications and port them to a mobile device. As mobile apps become more robust, this technique is somewhat lacking. A more sophisticated approach involves developing specifically for the mobile environment, taking advantage of both its limitations and advantages. For example, apps that use location-based features are inherently built from the ground up with an eye to mobile given that the user is not tied to a location, as on PC.

Apps are divided into two broad categories: native apps and web apps. Native apps are built for a specific mobile operating system, usually iOS or Android. Native apps enjoy better performance and a more finely-tuned user interface (UI), and usually need to pass a much stricter development and quality assurance process before they are released.

Web apps are used in HTML5 or CSS and require minimum device memory since they're run through a browser. The user is redirected on a specific web page, and all information is saved on a server-based database. Web apps require a stable connection to be used.

There are several types of apps currently available.

Gaming apps: The equivalent of computer video games, they are among the most popular types of apps. They account for one-third of all app downloads and three-fourths of all consumer spending.

Productivity apps: These focus on improving business efficiency by easing various tasks such as sending emails, tracking work progress, booking hotels, and much more.

Lifestyle and entertainment apps: Increasingly popular, these encompass many aspects of personal lifestyle and socialization such as dating, communicating on social media, as well as sharing (and watching) videos. Some of the most widely known apps such as Netflix, Facebook or TikTok fall into this category.

Other app types include mobile commerce (M-commerce) apps used to purchase goods online such as Amazon or eBay, travel apps that help a traveler in many ways (booking tours and tickets, finding their way through maps and geolocation, travel diaries, etc.), and utility apps such as health apps and barcode scanners.

KT2505 Internet of things

The internet of things is a technology that allows us to add a device to an inert object (for example: vehicles, plant electronic systems, roofs, lighting, etc.) that can measure environmental parameters, generate associated data and transmit them through a communications network.

What Is It?

The **internet of things (IoT)** is a set of technologies that uses sensors and actuators to inform us about the status of everyday items such as vehicles, tools and even living beings. It allows us to interact with them, enabling connectivity with platforms in the cloud that receive and process information for posterior analysis. This analyzed data is then used to make decisions.

Applications for Use

Here at Ferrovial, we mainly use IoT technology with connected cars, wearables, collaborative robotics, sensors in industrial plants (industry 4.0) and to efficiently manage energy.

We have developed a wide variety of projects in areas such as mobility, construction, highways, airports and industry. Some examples of initiatives carried out by our Digital Hub include:

Monitoring the Zity vehicle fleet, acquiring real-time telemetry and positioning data, carrying out bidirectional communication with drivers, and creating accident and harsh driving alerts — all to **reduce the occurrence of accidents**.