

Fundamental concepts of software and systems engineering

CPE3202 - Software and Systems Engineering

Dr. Pongrapee Kaewsaiha

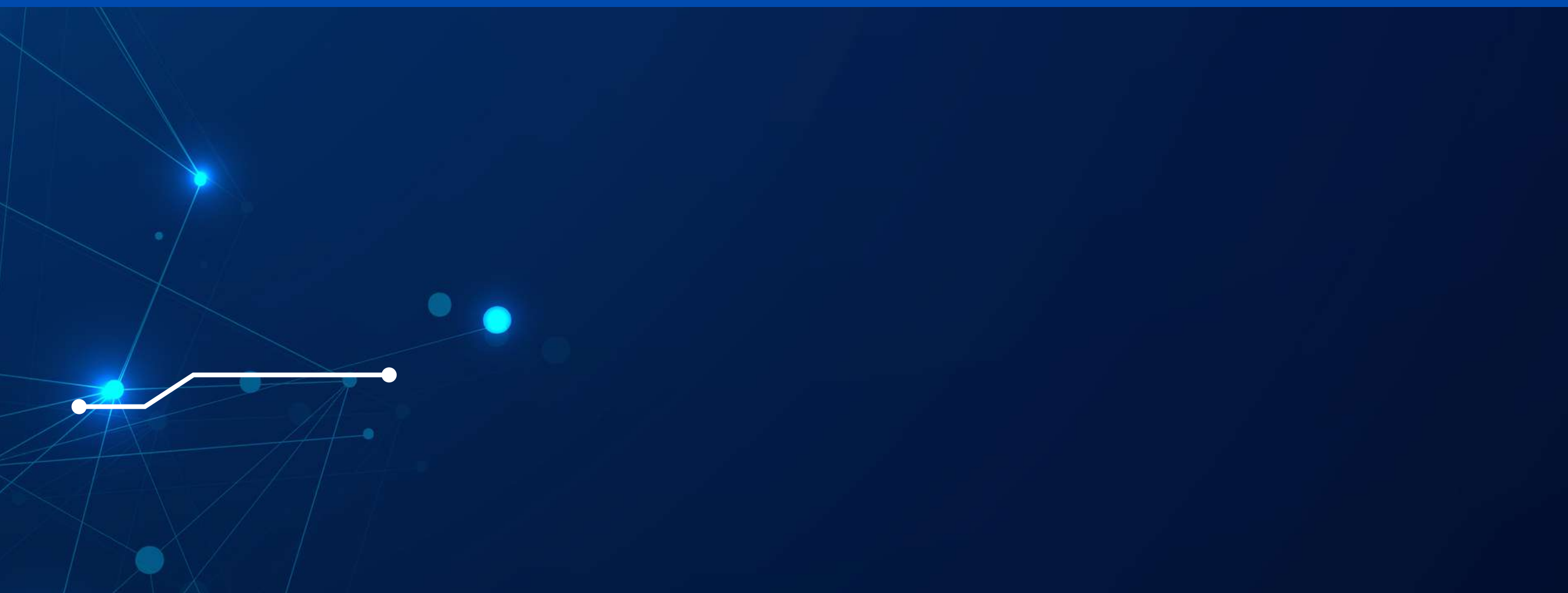


What is software & systems engineering?



Software system engineers use a systematic approach to:

1. Collect and analyze business and user requirements.
2. Design and create software to satisfy those requirements.
3. Test, launch, and maintain software and system.



History of software engineering



Computing began in the late 1950s but software engineering became a discipline in the 1960s. Software engineering transformed from ad hoc programming* towards more formal and standardized methods.

Software crisis

As people widely adopted computers, the inefficiencies in the software development made it difficult to meet the rapidly increasing demand. This led to the "Software Crisis" which began in the mid-1960s and lasted until the mid-1980s. During this period, software development often ran over budget, behind schedule, and consisted of buggy code.

The solution was to transform unorganized coding efforts into a well-established engineering discipline.

* Ad hoc programming = writing code in an unplanned way and for a particular purpose only.



Computer-aided software engineering (CASE)

CASE arised in the mid 1980s aiming to relieve the software crisis. CASE tools can be divided into six categories:

1. Business analysis and modeling
2. Development tools, such as debugging environments
3. Verification and validation tools
4. Configuration management
5. Metrics and measurement
6. Project management

Key terms

The following terms are used interchangeably:

- System engineers
- Software engineers
- Software developers

Each company will categorize software-related jobs using one of these combinations.

- Systems engineers & Software engineers
- Software engineers & Software developers

One will look at the holistic view while the other party will take care of technical approaches.



Systems engineers & Software engineers

Systems engineers	Software engineers
Manage engineering projects during their life cycle.	Design and develop good quality of software and applications.
Follow an interdisciplinary approach involving clients and users.	Follow a systematic and disciplined approach for software design, development, deployment and maintenance.
Require a broader background in Engineering, Mathematics, and Computer Science.	Require coding and database management skills.

Software engineers & Software developers

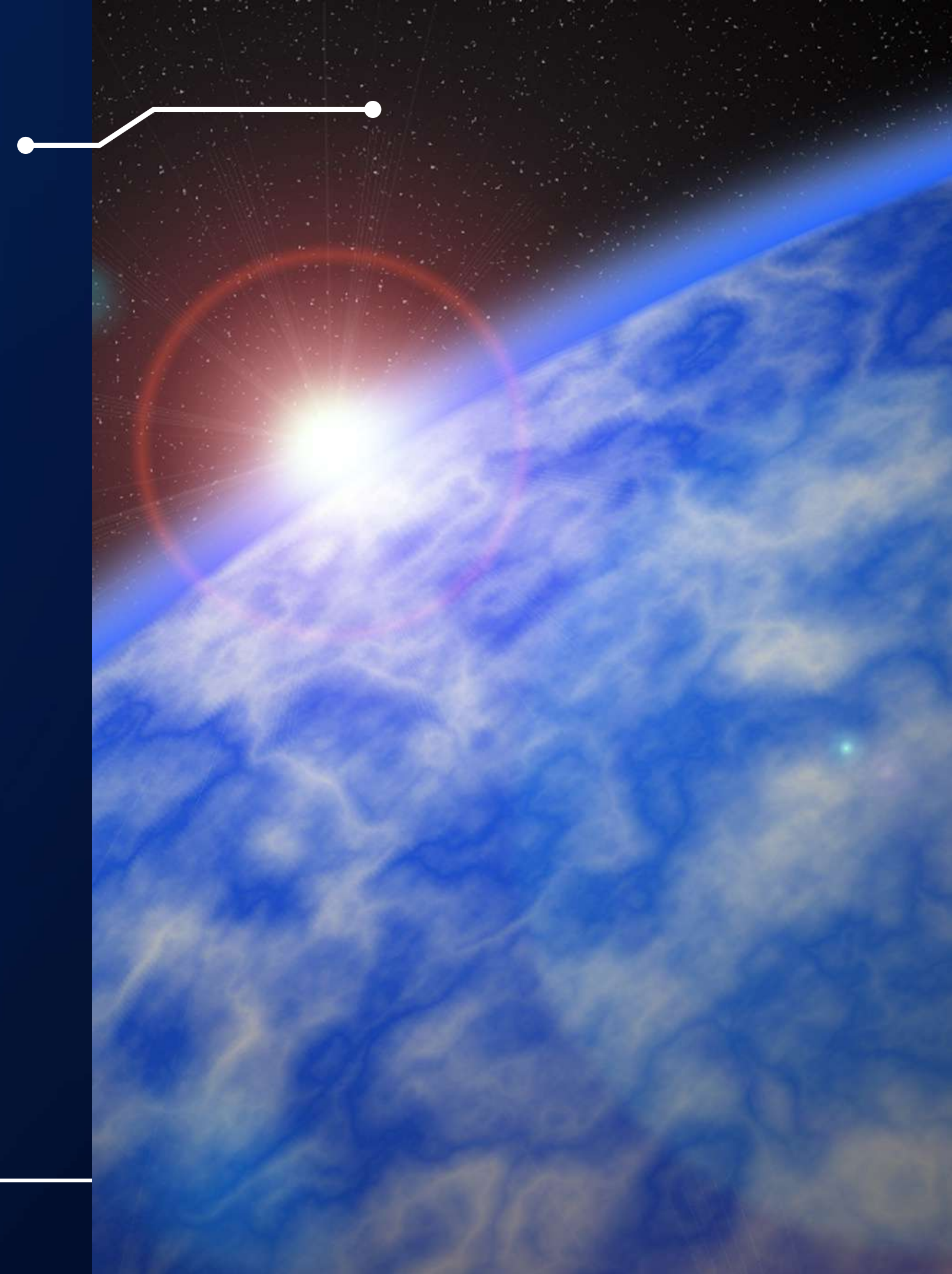
Software engineers	Software developers
Manage engineering projects during their life cycle.	Design and develop good quality of software and applications.
Follow an interdisciplinary approach involving clients and users.	Follow a systematic and disciplined approach for software design, development, deployment and maintenance.
Require a broader background in Engineering, Mathematics, and Computer Science.	Require coding and database management skills.

System development life cycle (SDLC)

- A systematic process to organize an engineering project (e.g., software development) in a predictable timeframe and budget to meet a client's requirements.
- Referred to as Software Development Life Cycle.

Six phases in the SDLC

1. Planning (*requirements, strategy, analysis*)
2. Design
3. Development (*building, implementation*)
4. Testing
5. Deployment (*launching, integration*)
6. Maintenance



SDLC

1. Planning

Requirements are gathered, analyzed, documented and prioritized. Write SRS.



2 Design

Develop system architecture based on SRS.



3. Development

Start the coding process (software development).



4. Testing

Test the system to ensure it is stable, secure, and meets the requirements in the SRS.



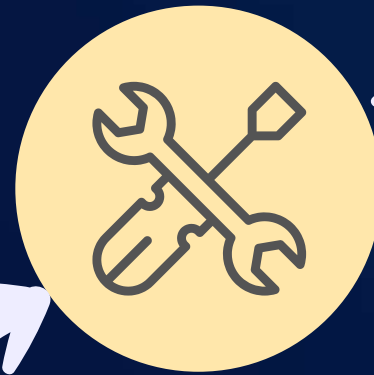
5. Deployment

Release the system into the production environment and made available to users.



6. Maintenance

Find bugs and other UI issues. Identify other requirements that may not listed in the SRS.



1. Planning

Requirements are gathered, analyzed, documented and prioritized. When planning a software solution, the following factors must be considered:



- Users of the solution
- Purpose of the solution
- Data inputs and outputs
- Legal and regulatory compliance
- Risk identification
- Quality assurance requirements
- Allocation of human and financial resources
- Project scheduling

Software/system requirements specification (SRS/SysRS)

1

Purpose

Definitions, background, system overview, references

2

Overall description

Product main functionalities, UI, design constraints

3

Specific requirements

Performance requirements, hardware, peripherals



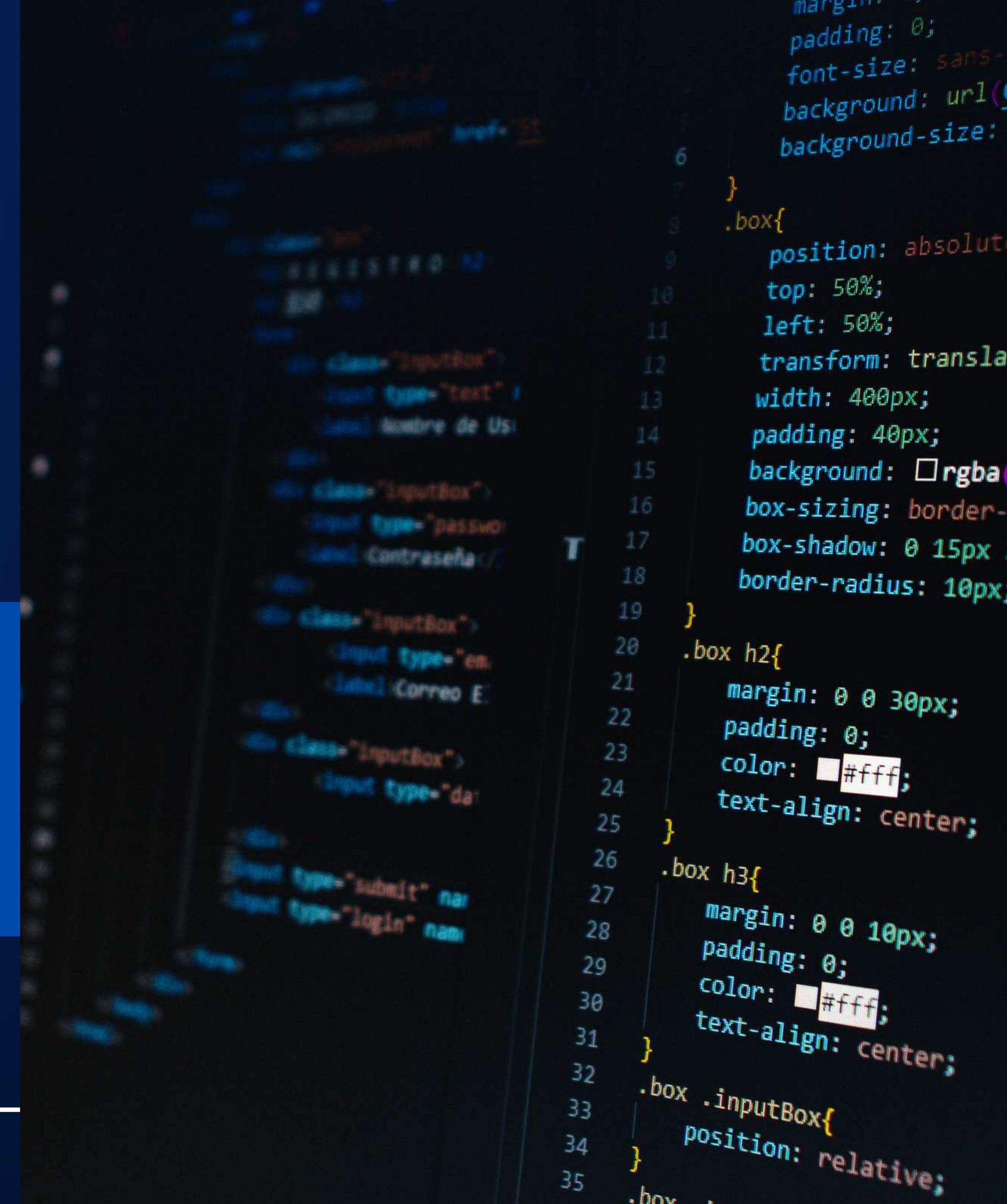
2. Design

- The requirements gathered from the SRS are used to develop the software architecture.
- Several team members work together to design the architecture.
- The architecture is reviewed by the stakeholders and team.
- A prototype can be designed as a preliminary mock-up of the system, or parts of the system, used for demonstration purposes.
- The document called a design document is created and used by developers during the next phase.



3. Development

- Sometimes called "building" or "implementation" phase.
- The developers start the coding process once the design document is completed.
- The project planners use the design document to determine and assign coding tasks.
- This phase often requires the use of programming tools, different programming languages, and software stacks.
- Organizations may have standards or guidelines that need to be followed.



4. Testing

- Code needs to be thoroughly tested to ensure it is stable, secure, and meets the requirements outlined in the SRS.
- Testing can be manual, automated, or a hybrid of both.
- Product bugs are reported, tracked, and fixed, and code is retested until the software is stable.
- Some common levels of testing include:
 - Unit testing - A module or component is tested by the developer.
 - Integration testing - A group of related modules are tested to verify they work together.
 - System testing - The application is tested as a whole.
 - User acceptance testing (UAT) - Tests carried out by selected testers or end-users.



5. Deployment

- Sometimes called "launching" or "integration" phase.
- Once the customer signs off on the functionality, it is released to production.
- This approach can be used for making software available on a website, mobile device app store, or other software distribution servers.



Releases

α

Alpha release

- May contain error
- Selected stakeholders

β

Beta release

- Meet requirements
- All stakeholders

GA

General availability

- Stable version
- All users

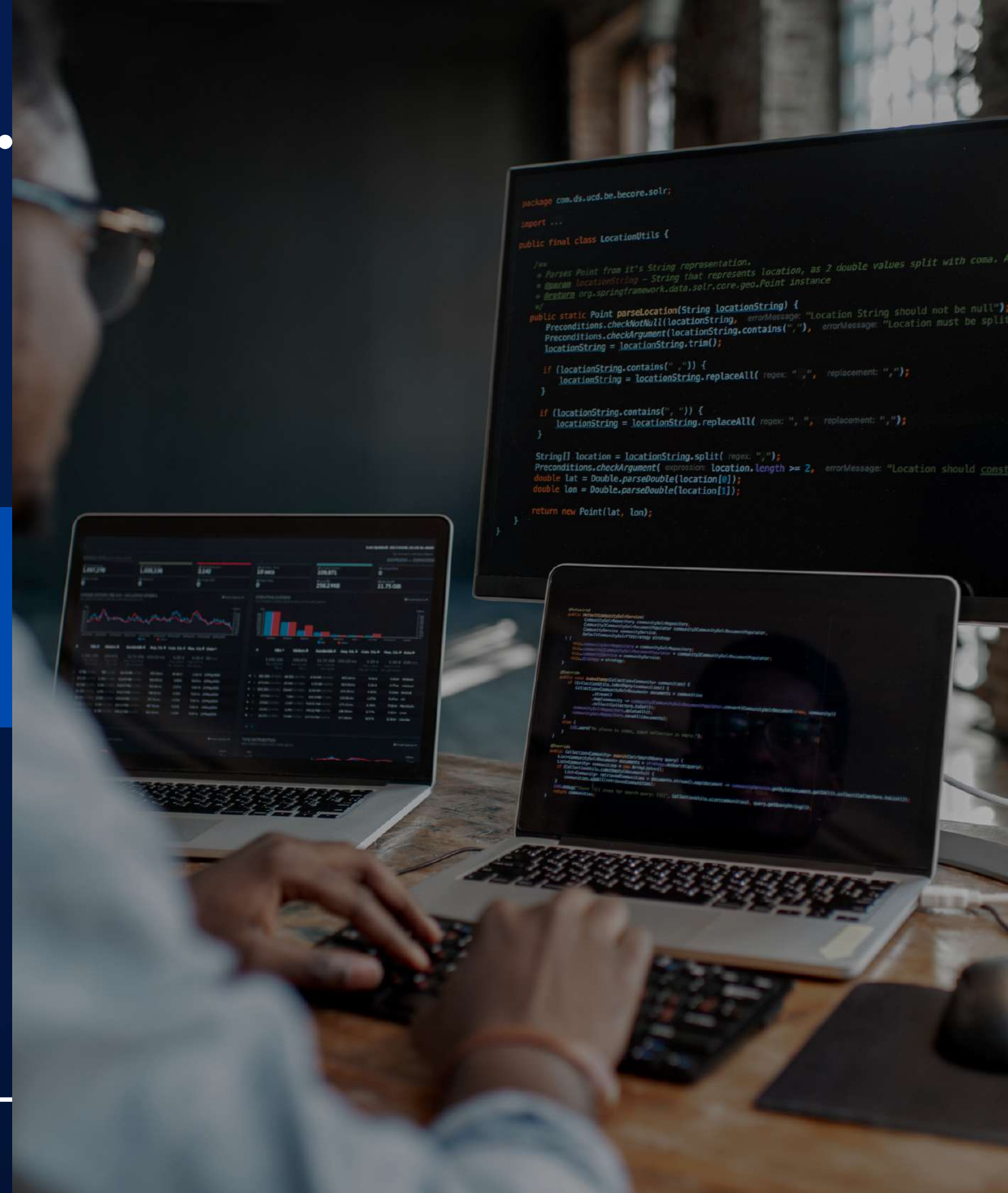


6. Maintenance

- Find any other bugs.
- Identify user interface (UI) issues.
- Identify other requirements that may not have been listed in the SRS.

These issues were missed during testing, so they need to be fixed or incorporated into the requirements of a future software release at the beginning of the next cycle.

- Code enhancement - Improve the quality, efficiency, or functionality of existing code without changing its behavior or purpose.





Key advantages of the SDLC



Gives development teams a process to follow (rather than using an ad hoc approach) to improve efficiency and reduce risks.




Offers an overview of the process, so stakeholders know where they fit in to that process.



Each phase is well defined so that team members know what they should be working on and when



Facilitates communication between the customer, other stakeholders, and the development team.





Key advantages of the SDLC



Each team member has a well-defined role which reduces conflict and overlapping responsibilities.



Provides room for iteration where, at the end of a cycle, the process can circle back to incorporate additional requirements as needed.



Problem are addressed in a timely manner in the design phase of the next cycle.

