**Module 6[[1]](#footnote-1)**

**SSRAI Software Development for HPC CI deployment**

**Outline:**

**Overview of Module**

* **Introduction**
* **Software Development**
* **Safe, Secure, and Reliable Software development**
* **Threats and vulnerabilities of AI software**
* **High Performance Robust Computing**

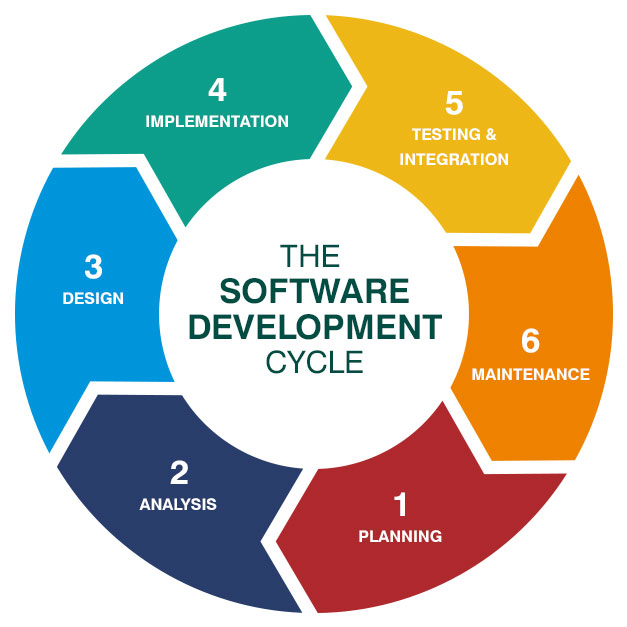
**1. Introduction**

This module discusses robust software development techniques for High Performance Computing (HPC) deployment of intelligent systems. The importance of adhering to good software development practices is magnified when addressing threats and vulnerabilities associated with intelligent systems because the risks posed by compromised A.I. are substantial. As High-Performance Computing (HPC) plays a key role in any area of computation, software development is no exception.

The goal of this module is to present the learner with techniques and concepts that will aid in the development of safe, secure, and reliable A.I. software. An emphasis is placed on the importance of secure development, the detection of flaws and issues, and the reduction of the probability of a system breach or collapse.

**2.** **Software Development**

IBM Research defines **software development**as a set of computer science activities dedicated to the process of creating, designing, deploying, and supporting software. The software development process can be split up in to 6 distinct stages. These stages are sometimes referred to as the **Software Development Life Cycle (SDLC).**



**Software Development Life Cycle**

*Step 1: Planning / Requirement Analysis*

Planning is arguably the most important phase of the SDLC. Creating a well-thought-out plan is fundamental to the process of creating safe, secure, and reliable software because it lays the groundwork for all other steps of the development life cycle. Planning is typically performed by a senior member of the software development team, preferably with input from all parties involved. This means the project lead should integrate information from management, marketing, sales, domain experts, engineers, customers, etc., into the design plan. For example, the ‘best’ software plan in the world developed without the context of a budget could easily turn out to be unfeasible because of financial constraints. The project lead should be able to design a plan that holds a holistic perspective of the project at hand.

*Step 2: Defining Requirements*

After the needs of the project have been clearly established, the next step involves clearly defining and documenting exactly what is expected of the software development team. After the requirements of the project have been documented through an SRS (Software Requirement Specification) document, they should be signed off by the customer or the market analysts depending on the nature of the project. The better the project lead is able to define what exactly is required / expected, the more likely the dev team will be able to deliver a quality product.

*Step 3: Designing the Product Architecture*

The SRS should be used by those involved in product architecture design to create multiple DDS (Design Document Specification) plans that will be presented to important stakeholders to review. The DDS should be measured on qualities like robustness, modularity, budget and time constraints, level of risk, and technical feasibility.

*Stage 4: Development*

The first three stages are all essentially planning stages with increasing levels of precision, or articulacy. In the fourth stage of the SDLC the product is built by a developer or team of developers per the selected DDS. If the first three stages are executed at a high level, this stage should be relatively easy, but if there isn’t a clearly defined, feasible design plan, the development team may struggle to deliver a suitable product.

*Stage 5: Product Testing*

This stage, as you may have guessed, is dedicated to the testing of the product prototype produced by the development team. Any issues with the product are documented, tracked, and reported back to the dev team so they can be fixed. This process is repeated until the product meets the SRS.

*Stage 6: Deployment and Maintenance*

After the product has been tested and revised thoroughly, the product is ready for deployment. Deployment may occur incrementally, or all at once, depending on the needs of the customer/user. Incremental deployments or beta testing allows for customer validation. Feedback from beta testing gives developers the chance to fix issues that might have been missed during the fifth stage. After the product is completely deployed, the product will be maintained for the user base. Periodic updates for bug fixes, general improvements, or new features are a necessary part of any sustained system.

Source: <https://bigwater.consulting/2019/04/08/software-development-life-cycle-sdlc/>

**Software Types**

Software is independent of hardware and makes computers programmable. Generally, there are three main types of software (See figure below):

1. **System software** provides the core functionality for a computer or system of computers. System software is composed of programs that manage the resources of the computer. Operating systems, database management systems, and software utilities are all examples of system software. System software is generally designed to control the hardware components of a computer, acting as a link between users and hardware. For example, the boot that loads operating system in RAM is a system software program.
2. **Programming software** (e.g., ‘programming language’) provides the tools needed for code developers and programmers. First, code is written in a high-level programming language like JAVA, C, and C++. After the program is written and implemented in the high-level language, programming software compiles that code in low-level language such as machine language.
3. **Application software** is software that performs specific tasks, typically for end-users. Every specific task that a computer can perform requires specific software designed for that purpose. The main difference between application software and system software is the presence of user interface. In system software, there are no user interfaces, however, application software is created to serve the users. In an application software, users can interact with software via graphical user interface (GUI). Famous examples of application software are Microsoft Office, web browsers (Mozilla Firefox, Internet Explorer) . Another example of application software are Integrated Development Environments or ‘IDEs’. IDEs provide comprehensive facilities to help developers build programs with programming software (which we considered as the second type of software). An IDE normally consists of source code editor (like sublime text), compiler (interpreter), build automation tools, and a debugger.

Diagram

Description automatically generated

Each type of software has its function and runs on the computer system and needs different types of software specialist. For instance, a music software development needs to be successful in notation and performance of music, or a developed medical software should meet key features of medical software requirements for healthcare facility. These show that software development is pervasive in many products from cars to house devices with a growing Internet of Things and that is why it is important.

**2.1. Key Features of Software Development**

**Artificial Intelligence:** AI systems that use neural networks, machine learning, natural language processing, and other cognitive-like capabilities help developers to offer innovative software development. AI-based software accelerates software deployment, quality, and efficacy. For example, IBM Watson allows developers to use AI services through application programming interfaces (API).

**Cloud-based development:** Software development organizations use cloud computing services to improve resource management and provide fast and cost-efficient IDE. Usually cloud-based development environments support all phases of the software development process (e.g., coding, designing, integrating, and testing).

**Blockchain**: Blockchain has extensive application in many areas because of its security features. Regardless of the industry or application, blockchain can transform the processes of many technologies. In software development area, the number of blockchain-oriented applications has increased due to its features that ensures the security of data. Blockchain offers tremendous opportunity for software development to change business operation.

**2.2 Software Development Tools and Solutions**

It is important to research which tools of software development are best for specific purposes of programming language, because the world of software development is changing so fast. There are many options that could be a challenge to select the best suitable one. The following list includes top popular software development tools that you may heard in some graduate courses:

1. **GitHub:** is the leading software development platform. GitHub provides an environment to store projects, codes, collaborative repository hosting service, and you have choice to make it public or private web-based service.
2. **Git:** is software for tracking changes in files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows.
3. **Jira:** is one of the first software development tool used by agile development by many organizations. Jira helps a lot when software development comes to project management, for example, for planning, tracking, customizing the workflow, and collaboration.
4. **Sublime Text:** is a text editor environment for code. Sublime allows you to change codes easily and switch between projects very fast.

1. **Slack:** is an environment that helps to share information, tools that are used between a group of people. Slack helps to reduce back and forth daily emails and increases the quality of communications between different teams of a company.

**3. Secure, Safe, Secure, and Reliable Software development**

**4. Threats and vulnerabilities of AI software**

Software vulnerabilities must be identified and prevented, which requires you to have an understanding of the vulnerability’s definition. What Causes Software Vulnerabilities? Top 10 Most Common Software Vulnerabilities How to Prevent Software Vulnerabilities

<https://technologyrivers.com/blog/how-to-find-and-mitigate-software-vulnerabilities/>

<https://www.infoworld.com/article/3607914/6-security-risks-in-software-development-and-how-to-address-them.html>

<https://dzone.com/articles/5-important-software-vulnerability-and-attacks-tha>

**5. High Performance Robust Computing**

1. This module can be integrated with CS 5750 Secure Software Development. [↑](#footnote-ref-1)