### Software Development Lifecycle (SDLC)

#### Understand and integrate security throughout the software development lifecycle (SDLC)

#### Development Methodologies

<table>
<thead>
<tr>
<th>Build and fix</th>
<th>Waterfall</th>
</tr>
</thead>
<tbody>
<tr>
<td>- No key architecture design</td>
<td>- Linear sequential lifecycle</td>
</tr>
<tr>
<td>- Problems fixed as they occur throughout the project</td>
<td>- Each phase is completed before moving on</td>
</tr>
<tr>
<td>- No formal development cycle</td>
<td>- No formal way to make changes during cycle</td>
</tr>
<tr>
<td>- Reactive, not proactive</td>
<td>- Project ends before collecting feedback and re-starting the cycle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prototyping</th>
<th>Incremental</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Rapid prototyping - quick sample to test the current project</td>
<td>- Iterative analysis and design are done early</td>
</tr>
<tr>
<td>- Evolutionary prototyping - incremental improvements to a design</td>
<td>- Easy to introduce new requirements</td>
</tr>
<tr>
<td>- Operational prototypes - incremental improvements intended for production</td>
<td>- Delivers incremental updates to software</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Spiral</th>
<th>Agile</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Iterative analysis and design</td>
<td>- Incremental development - quick development cycles</td>
</tr>
<tr>
<td>- Risk analysis during development</td>
<td>- Risk analysis during development</td>
</tr>
<tr>
<td>- Future information and requirements considered for risk analysis</td>
<td>- Allows for testing early in development</td>
</tr>
<tr>
<td>- Allows for testing early in development</td>
<td>- Highlights efficiency and iterative development</td>
</tr>
<tr>
<td>- Needs early user feedback</td>
<td>- User stories describe what a user does and why</td>
</tr>
<tr>
<td>- Prototypes filtered down to a few key features</td>
<td>- Prototypes filtered down to individual features</td>
</tr>
</tbody>
</table>

### Change Management Process

#### Request Control
- Develop organizational framework where users can request modifications, conduct cost/benefit analysis by management, and task prioritization by developers

#### Change Control
- Develop organizational framework where developers can create and test a solution before implementation in a production environment

#### Release Control
- Change approval before release

### Configuration Management Process

#### Software Version Control (SVC)
- A methodology for storing and tracking changes to software

#### Configuration Identification
- The labeling of software and hardware configurations with unique identifiers

#### Configuration Control
- Verify modifications to software versions with the change control and configuration management policies

#### Configuration Audit
- Ensure that the production environment is consistent with the accounting records

### Capability Maturity Model

#### Reactive
- 1. Initiating – informal processes, no formal way to make changes over time |
- 2. Repeatable – project management processes |
- 3. Defined – engineering processes, project planning, quality assurance, configuration management practices |
- 4. Managed – product and process improvement |
- 5. Optimizing – continuous process improvement |

### Project Management Tools

#### Gantt chart
- Type of bar chart that illustrates the relationship between tasks and activities|

#### Program Evaluation Review Technique (PERT)
- Project scheduling tool used to measure the capacity of a software product in development which uses to calculate risk

### Phases of object-oriented design

#### OOA (Requirements Analysis)
- Define classes of objects and interactions

#### OOD (Design)
- Objects are instances of classes

#### OOP (Programming)
- Introduce objects and methods

#### ORBs (Object Request Brokers)
- Work as mediators between the client and the server

#### CORBA (Common object request)
- Architecture and standards that use ORB to allow different systems and software on a system to interoperate with each other

#### Cohesion
- Work independently without help from other programs |
- High cohesion – No interaction with other classes |
- Low cohesion – Have interaction with other classes |
- Coupling - Level of interaction between objects

### Virus Types

#### Boot sector
- The most primitive form of a virus |

#### System infector
- Infects executable system files, BIOS and system commands

#### Steal
- Infects a systems factory installed UEFI (firmware)

#### Polymorphic
- The virus modifies the "garble" pattern as it spreads

#### Resident
- Loads as and when a program loads to the memory

### Anti-Virus Types

#### Signature based
- Not able to detect new malware or a.k.a. Zero day attacks

#### Heuristic based
- Static analysis without relying on signatures

### Protection Rings

#### Layer 0
- Operating system kernel

#### Layer 1
- Parts of the operating system other than the kernel

#### Layer 2
- I/O drivers and utilities

#### Layer 3
- Applications and programs