**Module 1: Inception**

* **Vision Document**: High-level description of the system's purpose and goals.
* **Use Case**: Describes interactions between actors and the system that yield value.
* **Scope Management**: Identifying and controlling project requirements to manage risk.
* **Business Case**: Justification for the project, including financial and non-financial benefits.

**Module 2: Elaboration**

* **Architecture Baseline**: A stable structure from which to proceed with detailed design.
* **Risk Assessment**: Identifying, analyzing, and managing risks.
* **Use Case Model**: Collection of use cases representing system functionality.
* **Iteration**: Small cycles of development to gradually refine the system.

**Module 3: Requirements Elicitation**

* **Stakeholders**: People with an interest in the system, including users and experts.
* **Interviews**: Elicitation technique where stakeholders provide requirements.
* **Prototyping**: Creating a working model to gather feedback on requirements.
* **JAD (Joint Application Development)**: Collaborative workshops for stakeholders to discuss requirements.

**Module 4: Requirements Analysis**

* **Data Models**: ER diagrams or class diagrams to represent data structures.
* **Behavioral Models**: Use case diagrams, sequence diagrams, or statecharts to show system behavior.
* **Flow Models**: Data flow diagrams to show the flow of data within the system.
* **4+1 View Model**: Architecture model with Logical, Process, Deployment, Implementation, and Use Case views.

**Module 5: Requirements Specification with Use Cases**

* **Use Case Diagram**: A visual representation of actors and their interactions with the system.
* **Happy Day Scenario**: The ideal flow where everything works as expected.
* **Preconditions**: Conditions that must be true for a use case to start.
* **Postconditions**: Results expected after a use case has been executed.

**Module 6: Use Case Analysis**

* **<<include>>**: A relationship where one use case includes the behavior of another.
* **<<extend>>**: A relationship where a use case can extend another for specialized behavior.
* **Generalization**: Use cases or actors that inherit behavior from others.
* **Analysis Class**: A class representing a key abstraction or responsibility within the system.

**Module 7: Activity Diagrams**

* **Action**: An atomic step in a workflow.
* **Activity**: A group of related actions.
* **Swimlane**: A partition of activities by who performs them (e.g., actors or systems).
* **Fork/Join**: Represents concurrency in an activity diagram where multiple actions occur simultaneously.

**Module 8: Sequence Diagrams**

* **Lifeline**: Represents the lifecycle of an object or actor in a sequence diagram.
* **Synchronous Message**: A message where the sender waits for a response.
* **Asynchronous Message**: A message where the sender does not wait for a response.
* **Loop**: A repeated interaction between objects.

**Module 9: Requirements Quality**

* **Correctness**: Ensuring all requirements are accurate and meet user needs.
* **Completeness**: All necessary requirements are included.
* **Unambiguous**: Requirements can only be interpreted one way.
* **Verifiability**: A requirement is testable to ensure it's implemented correctly.
* **Traceability**: Ensures each requirement can be traced to design, code, and tests.

**Module 10: Requirements Standards**

* **IEEE 29148**: Modern IEEE standard for software requirements, covering process, elicitation, and management.
* **CMMI**: Capability Maturity Model Integration, a process improvement framework that includes requirements development and management.
* **SWEBOK**: Software Engineering Body of Knowledge, a comprehensive guide to software development, including requirements engineering.
* **DO-178C**: Avionics software standard that mandates rigorous traceability and verification.

**Module 11: Requirements Decomposition**

* **Flow-Down**: Assigning high-level requirements to subsystems.
* **Refinement**: Breaking down requirements into actionable details for implementation.
* **Completion**: Ensuring all code is traced to requirements, particularly in safety-critical industries.
* **Derived Requirements**: Requirements inferred or added during refinement or decomposition.

**Module 12: Requirements Management**

* **Traceability**: Linking requirements to project elements like design, tests, and code.
* **Change Control Board (CCB)**: A committee that manages and approves changes to requirements.
* **Change Request Management (CRM)**: The process of handling changes to requirements.
* **Requirements Maturity Levels**: Stages of requirements management from chaos (no management) to integrated (fully integrated requirements management).