Learning Report - Embedded Software Design, Development Processes and Standard

GLOBAL ENGINEERING ACADEMY

Genesis

L&T Technology Services

Document History

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	Approved By	Remarks/Revision Details
1	10/10/2020	Vinay B J		Bhargav N	Added Activity-1, 2 & 3

Contents

Table of Figures

Table of Tables

Activity 1: White, Black, Grey box testing and its applications

White Box Testing

White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.

It is also called glass box testing or clear box testing or structural testing.

Working process of white box testing:

- Input: Requirements, Functional specifications, design documents, source code.
- Processing: Performing risk analysis for guiding through the entire process.
- Proper test planning: Designing test cases so as to cover entire code. Execute rinse-repeat until error-free software is reached. Also, the results are communicated.
- Output: Preparing final report of the entire testing process.

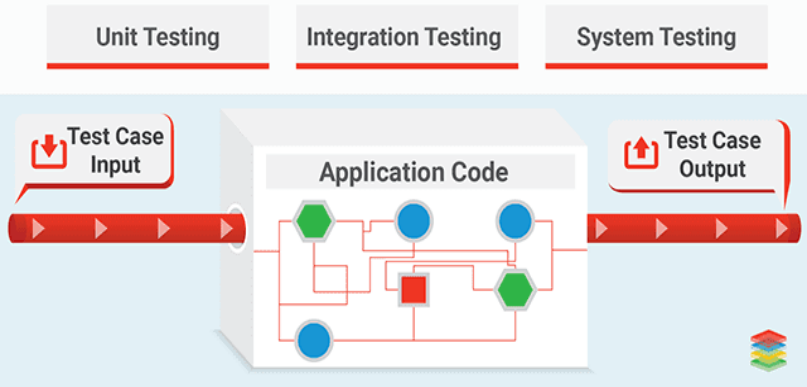

Figure 1: White Box Testing Approach

Reasons for white box testing:

- It identifies internal security holes.
- To check the way of input inside the code.
- Check the functionality of conditional loops.
- To test function, object, and statement at an individual level.

Types of White Box Testing:

- Path testing: In the path testing, we will write the flow graphs and test all independent paths.
- Loop testing: In the loop testing, we will test the loops such as while, for, and do-while, etc. and also check for ending condition if working correctly and if the size of the conditions is enough.
- Condition testing: In this, we will test all logical conditions for both true and false values; that is, we will verify for both if and else condition.

Advantages of White box testing:

- White box testing optimizes code so hidden errors can be identified.
- Test cases of white box testing can be easily automated.
- This testing is more thorough than other testing approaches as it covers all code paths.
- It can be started in the SDLC phase even without GUI.

Disadvantages of White box testing:

- White box testing is too much time consuming when it comes to large-scale programming applications.
- White box testing is much expensive and complex.
- It can lead to production error because it is not detailed by the developers.
- White box testing needs professional programmers who have a detailed knowledge and understanding of programming language and implementation.

Applications of White Box Testing:

- Security gaps and vulnerabilities - checking to see if security best practices were applied when coding the application, and if the code is vulnerable to known security threats and exploits.
- Broken or poorly structured paths - identifying conditional logic that is redundant, broken or inefficient.
- Expected output - executing all possible inputs to a function to see if it always returns the expected result.
- Loop testing - checking single loops, concatenated loops and nested loops for efficiency, conditional logic, and correct handling of local and global variables.
- Data Flow Testing (DFT) - tracking variables and their values as they pass through the code to find variables that are not correctly initialized, declared but never used, or incorrectly manipulated.

Black Box Testing

Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester.

Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

Working process of black box testing:

- Requirement: This is the initial stage of SDLC and in this stage, a requirement is gathered. Software testers also take part in this stage.
- Test Planning & Analysis: Testing Types applicable to the project are determined. A Test Plan is created which determines possible project risks and their mitigation.
- Design: In this stage Test cases/scripts are created on the basis of software requirement documents
- Test Execution: In this stage Test Cases prepared are executed. Bugs if any are fixed and re-tested.

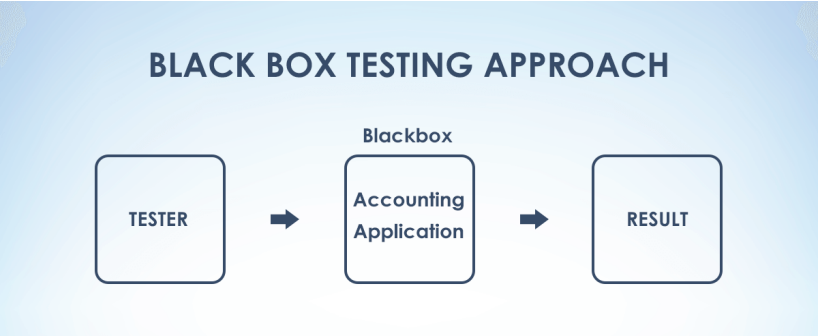

Figure 2: Black Box Testing Approach

Types of Black Box Testing:

- Functional testing: This black box testing type is related to the functional requirements of a system; it is done by software testers.
- Non-functional testing: This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.
- Regression testing: Regression Testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

Advantages of Black box testing:

- Well suited and efficient for large code segments.
- Code access is not required.
- Clearly separates user's perspective from the developer's perspective through visibly defined roles.
- Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language, or operating systems.

Disadvantages of Black box testing:

- Limited coverage, since only a selected number of test scenarios is actually performed.
- Inefficient testing, due to the fact that the tester only has limited knowledge about an application.
- Blind coverage, since the tester cannot target specific code segments or error-prone areas.
- The test cases are difficult to design.

Applications of Black Box Testing:

- Black box test evaluates all relevant subsystems, including UI/UX, web server or application server, database, dependencies, and integrated systems.
- Security technology that performs black box testing is Dynamic Application Security Testing (DAST), which tests products in staging or production and provides feedback on compliance and security issues.

Gray Box Testing

Gray Box Testing is a software testing technique which is a combination of Black Box Testing technique and White Box Testing technique. In Black Box Testing technique, tester is unknown to the internal structure of the item being tested and in White Box Testing the internal structure is known to tester. The internal structure is partially known in Gray Box Testing. This includes access to internal data structures and algorithms for purpose of designing the test cases.

Techniques used for Grey box Testing are:

- Matrix Testing: This testing technique involves defining all the variables that exist in their programs.
- Regression Testing**:** To check whether the change in the previous version has regressed other aspects of the program in the new version. It will be done by testing strategies like retest all, retest risky use cases, retest within a firewall.
- Orthogonal Array Testing or OAT**:** It provides maximum code coverage with minimum test cases.
- Pattern Testing: This testing is performed on the historical data of the previous system defects. Unlike black box testing, gray box testing digs within the code and determines why the failure happened.

Figure 3: Gray Box Testing Approach

Steps to perform Gray Box Testing

- Identify inputs
- Identify the outputs
- Identify the major paths
- Identify Sub functions
- Develop inputs for Sub functions
- Develop outputs for Sub functions
- Execute test case for Sub functions
- Verify the correct result for Sub functions
- Repeat steps 4 & 8 for other Sub functions
- Repeat steps 7 & 8 for other Sub functions

Advantages of Gray Box Testing:

- Users and developers have clear goals while doing testing.
- Gray box testing is mostly done by the user perspective.
- Testers are not required to have high programming skills for this testing.
- Gray box testing is non-intrusive.
- Overall quality of the product is improved.
- In Gray box testing, developers have more time for defect fixing.
- By doing Gray box testing, benefits of both black box and white box testing is obtained.
- Gray box testing is unbiased. It avoids conflicts between a tester and a developer.
- Gray box testing is much more effective in integration testing.

Disadvantages of Gray box testing:

- Defect association is difficult when Gray testing is performed for distributed systems.
- Limited access to internal structure leads to limited access for code path traversal.
- Because source code cannot be accessed, doing complete white box testing is not possible.
- Gray box testing is not suitable for algorithm testing.
- Most of the test cases are difficult to design.

Applications of Gray Box Testing:

- Grey-box testing is a perfect fit for Web-based applications.
- Grey-box testing is also a best approach for functional or domain testing.

Difference between Black Box, White Box and Gray Box Testing

Black Box Testing	White Box Testing	Gray Box Testing
The internal workings of an application need not be known.	The tester has limited knowledge of the internal workings of the application.	Tester has full knowledge of the internal workings of the application.
Also known as closed-box testing, data-driven testing, or functional testing.	Also known as translucent testing, as the tester has limited knowledge of the insides of the application.	Also known as clear-box testing, structural testing, or code-based testing.
Performed by end-users and also by testers and developers.	Performed by end-users and also by testers and developers.	Normally done by testers and developers.
Testing is based on external expectations - Internal behavior of the application is unknown.	Testing is done on the basis of high-level database diagrams and data flow diagrams.	Internal workings are fully known and the tester can design test data accordingly.
Not suited for algorithm testing.	Not suited for algorithm testing.	Suited for algorithm testing.
This can only be done by trial-and-error method.	Data domains and internal boundaries can be tested, if known.	Data domains and internal boundaries can be better tested.
It is the least time-consuming process among all the testing processes.	The entire testing process is the most time consuming among all the testing processes.	It is less time consuming than White Box testing.
It is less exhaustive than White Box and Grey Box testing methods.	It is most exhaustive than Black Box and Gray Box testing methods.	Partly exhaustive; depends upon the type of test cases are coding based or GUI based.
The base of this testing is external expectations internal behavior is unknown.	The base of this testing is coding which is responsible for internal working.	Testing based on high-level database diagrams and dataflow diagrams.

Table 1: Difference Between Black, White and Gray Box Testing

Activity 2: Various Standards and Tools

Aerospace Industry

AS9100 is the Quality Management System Requirements for Aerospace Manufacturers, but it also references several other standards that an organization must understand and integrate into their QMS.

Critical Standards for AS9100 Implementation:

- **ARP 9134A – Supply Chain Risk Management.** The guideline focuses on Quality as a key risk assessment factor taking into account elements from all aspects of the business having a direct link to global quality management.
- **AS5553C – Counterfeit Electronic Parts.** This standard is for use by organizations that procure and/or integrate EEE parts and/or assemblies containing such items.
- **AS9015A – Work Transfer Supplier Self Verification Process.** This standard defines the requirements for a delegation process identified in Aerospace Quality Management System standards.
- **AS9101 – QMS Audit Requirements**. This standard defines requirements for the preparation and execution of the audit process.
- **AS9102 – First Article Inspection.** This standard establishes the baseline requirements for performing and documenting FAI.
- **AS9103A – Key Characteristics**. This standard is primarily intended to apply to new parts and products, but can also be applied to parts currently in production.
- **AS9131C – Nonconformance.** This standard defines the common nonconformance data definition and documentation that must be exchanged between an internal/external supplier or sub-tier supplier, and the customer when informing about a nonconformity requiring formal decision.
- **AS9146 – Foreign Object Detection (FOD).** This standard defines FOD Prevention Program requirements for organizations that design, develop, and provide aviation, space, and defense products and services. And by organizations providing post-delivery support, including the provision of maintenance, spare parts, or materials for their own products and services.
- **ISO 10006 – Project Management.** This standard gives guidelines for the application of quality management in projects.
- **ISO 10007 – Configuration Management.** This standard provides guidance on the use of configuration management within an organization.
- **ISO 19011 – Guidelines for Auditing Management Systems.** This document provides guidance on auditing management systems.

AS9100 Core Tools.

- **ARP9013-1A –** Statistical Product Acceptance Requirements Using Isolated Lot Sampling Methods
- **ARP9013-2A –** Statistical Product Acceptance Requirements Using Attribute Or Variable Lot Acceptance Sampling Plans
- **ARP9013-3A –** Statistical Product Acceptance Requirements Using Process Control Methods

- **ARP9013-4A –** Statistical Product Acceptance Requirements Using Continuous Sampling, Skip-Lot Sampling, Or Methods for Special Cases
- **AS9145 –** Requirements for Advanced Product Quality Planning and Production Part Approval Process
- **ARP5580 –** Recommended Failure Modes and Effects Analysis (FMEA) Practices
- **ARP9136 –** Root Cause Analysis and Problem Solving
- **AS13004-** Process Failure Mode and Effects Analysis (PFMEA) and Control Plans

Aerospace Software

- **AS9115A** – Deliverable Aerospace Software Supplement for AS9100C (Supersedes AS9006:2003)
- **AS9006A** – Deliverable Aerospace Software Supplement for AS9100A
- **ARP9005B** – Aerospace Guidance for Non-Deliverable Software
- **ARP9034A** – A Process Standard for the Storage, Retrieval, and Use of 3D Type Design Data

Aerospace Supply Chain

- **ARP900B** – Aerospace Contract Clauses – used when subcontracting and purchasing from 3rd parties
- **ARP9107A** – Direct Delivery Authorization Guidance for Aerospace Companies
- **ARP9114A** – Direct Ship Guidance for Aerospace Companies
- **ARP9134A** – Supply Chain Risk Management Guidelines

Railway Industry

The following codes, standards and specifications applies to all systems and equipment, as the case maybe, forming part of the project:

- **NFPA 130:** Standard for Fixed Guideway Transit and Passenger Rail Systems: This standard specifies fire protection and life safety requirements for underground, surface, and elevated fixed guideway transit and passenger rail systems.
- **BS EN 50126:** Railway applications-The specification demonstrate reliability, availability, maintainability and safety (RAMS). This is a multi-part standards divided into many parts such as: o BS EN 50126-1: Railway applications. The specification and demonstration of reliability, availability, maintainability and safety (RAMS). Basic requirements and generic process
- **EN 50119** Railway applications - Fixed installations - Electric traction overhead contact lines
- **EN 50121**: Railway applications- Electromagnetic compatibility: This is a multi-part document divided into the following parts: o EN 50121-1: Railway applications. Electromagnetic compatibility. General
- **EN 50121-2:** Railway applications. Electromagnetic compatibility. Emission of the whole railway system to the outside world o EN 50121-3-1: Railway applications. Electromagnetic compatibility. Rolling stock. Train and complete vehicle
- **EN 50121-3-2:** Railway applications. Electromagnetic compatibility. Rolling stock. Apparatus o EN 50121-4: Railway applications. Electromagnetic compatibility. Emission and immunity of the signalling and telecommunications apparatus

- **EN 50121-5**: Railway applications. Electromagnetic compatibility. Emission and immunity of fixed power supply installations and apparatus.
- **BS EN 50122**: Railway applications, Fixed installations, Electrical safety, earthling and the return circuit: This is a multi-part document divided into the following parts:
- **BS EN 50122-1** Railway applications. Fixed installations. Protective provisions relating to electrical safety and earthing
- **BS EN 50122-2** Railway applications. Fixed installations. Protective provisions against the effects of stray currents caused by d.c. traction systems o BS EN 50122-3 Railway applications. Fixed installations. Electrical safety, earthing and the return circuit. Mutual Interaction of a.c. and d.c. traction systems
- **EN 50163:** RAILWAY APPLICATIONS - SUPPLY VOLTAGES OF TRACTION SYSTEMS: This European Standard specifies the main characteristics of the supply voltages of traction systems, such as traction fixed installations, including auxiliary devices fed by the contact line, and rolling stock, for use in the applications such as Railways, Guided mass transport systems such as tramways, elevated and underground railways mountain railways, and trolleybus systems and Material transportation systems.
- **IEC 60364 (4-41):** Electric installation of Buildings - Electric Shocks having following as applicable releases. o IEC 60364-4-41 Amd.1 Ed. 5.0 b:2017 (Amendment 1 - Low voltage electrical installations - Part 4-41: Protection for safety - Protection against electric shock) o IEC 60364-4-41 Ed. 5.0 b:2005 o IEC 60364-4-41 Ed. 5.1 b:2017
- **IEEE 80-2013:** Guide for Safety in AC Substation Grounding: This guide is primarily concerned with outdoor ac substations, either conventional or gas-insulated. These include distribution, transmission, and generating plant substations.
- **IEEE 519-2014:** Recommended Practice and Requirements for Harmonic Control in Electric Power Systems: Goals for the design of electrical systems that include both linear and nonlinear loads are established in this recommended practice.
- **IS 1893:** CRITERIA FOR EARTHQUAKE RESISTANT DESIGN OF STRUCTURES

Automobile Industry

- **ISO 14001**: As the international standard for environmental management systems — or EMS — ISO 14001 is the primary EMS certification for more than 250,000 organizations around the world. As the global standard for any business that wants to manage and positively control all aspects of its environmental impact, ISO 14001 certification is a proven way to demonstrate that you're serious about your business's environmental and economic sustainability.
- **ISO 45001:** Along with offering reliable products and services, automotive manufacturers must constantly strive to provide their workers and visitors with a safe and healthy business environment. With the ultimate goal of providing businesses with a framework for controlling and eliminating factors that can lead to illness, injuries and — in worst-case scenarios — death, obtaining ISO 45001 Health and Safety certification is a prudent move for any organization's senior management to support.

- **IATF 16949:** We work with the automotive industry to support the manufacturing of safe and reliable products, which are produced and continually improved to meet or exceed customer and regulatory authority requirements. Most organizations manufacturing for the automotive industry are required to be certificated to IATF 16949, which was developed by the International Automotive Task Force (IATF).
- **ISO 9001**: Since IATF 16949 isn't designed to be a self-sufficient quality standard but instead works best in conjunction with a comprehensive QMS, ISO 9001 certification makes a great deal of sense for automotive companies looking to demonstrate improvement in customer satisfaction, operating costs, stakeholder relationships, legal compliance, risk management, business credentials and attracting new business.
- **AIS-100:** Pedestrian protection
- **AIS-99:** Side mobile deformable offset
- **AIS-098:** Offset frontal crash
- **AUTOSAR:** It pursues the objective to create and establish an open and standardized software architecture for automotive electronic control units (ECUs).
- **ASIL-D:** An Automotive Safety Integrity Level (ASIL)-decomposed (or SIL-synthesized) architecture offers a reliable and robust path to achieving the highest levels of diagnostic coverage and gives end-equipment designers increased flexibility when developing safety-critical systems with high ASIL or SIL requirements.
- **ASPICE:** The Automotive SPICE Process Assessment Model (PAM) has been developed by consensus of the car manufacturers within the Automotive Special Interest Group (SIG) of the joint Procurement Forum / SPICE User Group under the Automotive SPICE initiative.
- **ISO26262:** It defines functional safety for automotive equipment applicable throughout the lifecycle of all automotive electronic and electrical safety-related systems.

Activity 3: Overview of diagrams used in SysML

- A graphical modelling language developed in response to the UML for Systems Engineering RFP developed by the OMG, INCOSE, and AP233a
- Supports the specification, analysis, design, verification, and validation of systems that include hardware, software, data, personnel, procedures, and facilities
- It is a visual modeling language that provides – Semantics = meaning, connected to a metamodel (rules governing the creation and the structure of models) – Notation = representation of meaning, graphical or textual.

SysML vs UML
UML is a general-purpose graphical modeling language aimed at Software Engineers

- Object diagram,
- Deployment diagram,
- Component diagram,
- Communication diagram,
- Timing diagram and
- Interaction overview diagram

Diagrams from UML

- Class diagram (Block Definition Diagram - Class→ Block)
- Package diagram
- Composite Structure diagram (Internal Block Diagram)
- State Machine Diagram
- Activity Diagram
- Use Case Diagram
- Sequence Diagram

In addition, SysML adds some new diagrams and constructs

- Parametric diagram,
- Requirement diagram
- Flow ports,
- Flow specifications
- Item flows.
- Allocation

SysML Extensions

- Blocks
- Item flows
- Value properties
- Allocations
- Requirements
- Parametrics
- Continuous flows

Available Diagrams

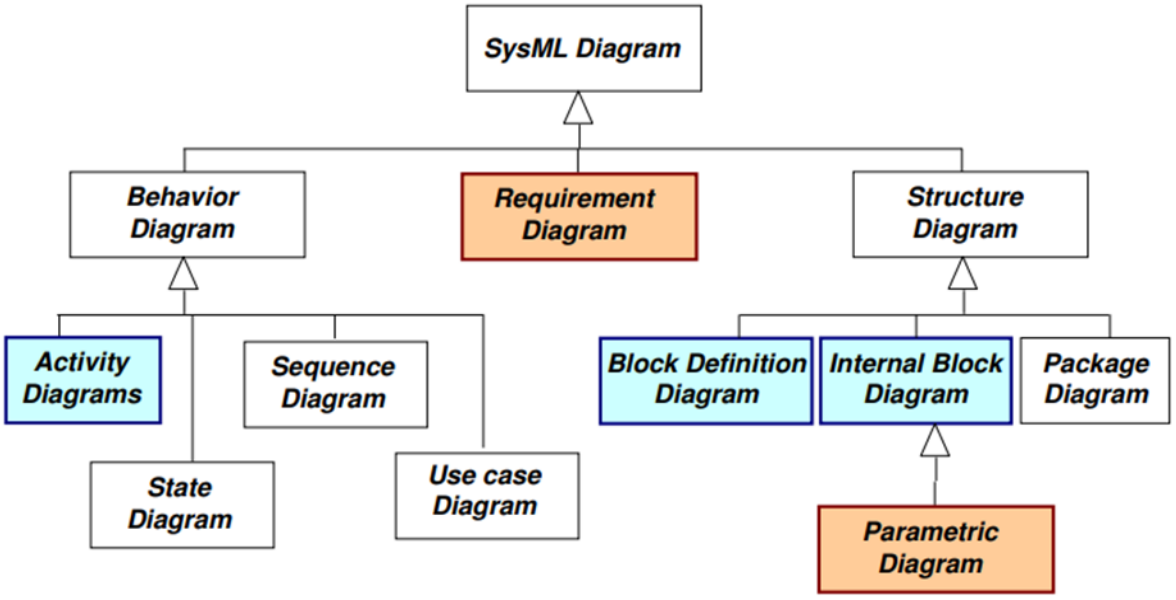

Figure 4: SysML Diagrams Overview

Behavior Diagram

The behavior diagrams include the use case diagram, activity diagram, sequence diagram, and state machine diagram.

Activity Diagram

The purpose of Activity diagrams is to specify dynamic system behaviors that Satisfy System Functional Requirements using both Control and Object (data) Flows. When properly applied Activity diagrams are recursively scalable. An Activity diagram shows system dynamic behavior using a combined Control Flow and Object (data) Flow model.

Sequence diagram

A Sequence diagram is a dynamic behavioral diagram that shows interactions (collaborations) among distributed objects or services via sequences of messages exchanged, along with corresponding (optional) events. The purpose of Sequence diagrams is to specify dynamic system behaviors as message-passing collaborations among prototypical Blocks (Parts). When properly applied (See Usage Notes below) Activity diagrams are recursively scalable and simulatable.

State Machine diagram

A State Machine diagram is a dynamic behavioral diagram that shows the sequences of States that an object or an interaction go through during its lifetime in response to Events (a.k.a. "Triggers"), which may result in side-effects (Actions). The purpose of State Machine diagrams is to specify dynamic system behaviors for time-critical, mission-critical, safety-critical, or financially-critical objects. When properly applied (See Usage Notes below) State Machine diagrams are recursively scalable and simulatable.

Use Case diagram

A Use Case diagram shows communications among system transactions (Use Cases) and external users (Actors) in the context of a system boundary (Subject; notation: rectangle). Actors may represent wetware (persons, organizations, facilities), software systems, or hardware systems. Defining relationships between the system Subject and the system Actors is an effective informal way to define system scope. The purpose of Use Case diagrams is to provide a high-level view of the subject system and convey the top-level system requirements in non-technical terms for all stakeholders, including customers and project managers as well as architects and engineers. Additional more rigorous SysML diagrams are needed to specify a scalable and simulatable System Architecture Model (SAM).

Requirement diagram

SysML predefines the following stereotype specializations of NFRs:

- Performance Requirement
- interface Requirement
- design Constraint
- physical Requirement

A SysML Requirement diagram is a static structural diagram that shows the relationships among Requirement constructs, model elements that Satisfy them, and Test Cases that Verify them.
The purpose of Requirement diagrams is to specify both Functional and Non-Functional Requirements within the model so that they can be traced to other model elements that Satisfy them and Test Cases that Verify them.

Structure Diagram

Block Definition Diagram

A Block Definition Diagram is a static structural diagram that shows system components, their contents (Properties, Behaviors, Constraints), Interfaces, and relationships.
Blocks can be recursively decomposed ("nested") into Parts by alternating between Block Definition Diagram (BDD) definitions and Internal Block Diagram (IBD) usages
Behaviors can either be encapsulated by Blocks (e.g., Operations, Signals, and State Machines) or Allocated (via «allocate» Dependency) to Blocks (e.g., Activities/Actions) directly or indirectly (via Interfaces).
Blocks can be mathematically constrained via Constraint Blocks to produce mathematically simulatable Parametric diagrams.
compare and contrast: UML 2 Class and Component diagrams; SA/SD System Context & Structure Chart diagrams; IDEF IDEF1X diagrams.

Purpose:

The purpose of Block Definition Diagrams is to specify system static structures that be used for Control Objects, Data Objects, and Interface Objects. When properly applied (See Usage Notes below) Block diagrams are recursively scalable and mathematically (parametrically) simulatable

Internal Block Diagram

An Internal Block Diagram is a static structural diagram owned by a particular Block that shows its encapsulated structural contents: Parts, Properties, Connectors, Ports, and Interfaces. Stated otherwise, an IBD is a "white-box" perspective of an encapsulated ("black-box") Block.
Blocks can be recursively decomposed ("nested") into Parts by alternating between Block Definition Diagram (BDD) definitions and Internal Block Diagram (IBD) usages
Behaviors can either be encapsulated by Blocks (e.g., Operations, Signals, and State Machines) or Allocated (via «allocate» Dependency) to Blocks (e.g., Activities/Actions) directly or indirectly (via Interfaces).

Blocks can be mathematically constrained via Constraint Blocks to produce mathematically simulatable Parametric diagrams.

compare and contrast: UML 2 Class and Component diagrams; SA/SD System Context & Structure Chart diagrams; IDEF IDEF1X diagrams.

Purpose:

The purpose of Internal Block Diagrams (IBDs) is to show the encapsulated structural contents (Parts, Properties, Connectors, Ports, Interfaces) of Blocks so that they can be recursively decomposed and "wired" using Interface Based Design techniques. When used correctly BDDs + IBDs are recursively scalable and mathematically (parametrically) simulatable

Parametric diagram

A Parametric diagram is a specialization of an Internal Block Diagram (IBD) that enforces mathematical rules (Constraints) defined by Constraint Blocks across the internal Part Value Properties bound by the Constraint Block Parameters.

Binding Connectors (keyword = «equal») between Constraint Block Parameters and internal Part Value Properties effect constraint satisfaction (propagation)

Purpose:

The purpose of Parametric diagrams (PARs) is to enforce mathematical rules across Block Value Properties. When used correctly BDDs + IBDs + PARs are recursively scalable and mathematically simulatable.

Package diagram

A Package diagram is a static structural diagram that shows the relationships among packages and their contents. Package can be stereotyped (customized) for organizing model elements into models, views, model libraries, and frameworks.

Purpose:

The purpose of Package diagram is to support the organization and management of large, complex System Architecture Models (SAMs).

References

- https://www.geeksforgeeks.org/software-engineering-white-box-testing/
- https://www.javatpoint.com/white-box-testing
- https://www.geeksforgeeks.org/software-engineering-black-box-testing/
- https://www.guru99.com/black-box-testing.html
- https://www.tutorialspoint.com/software_testing/software_testing_methods.htm
- https://www.geeksforgeeks.org/gray-box-testing-software-testing/
- https://www.guru99.com/grey-box-testing.html
- https://www.javatpoint.com/black-box-testing-vs-white-box-testing-vs-grey-box-testing
- https://www.tutorialspoint.com/software_testing/software_testing_methods.htm
- https://as9100store.com/aerospace-standards-explained/
- www.diva-portal.org/smash/get/diva2:430399/fulltext01.pdf
- inf.mit.bme.hu/sites/default/files/materials/taxonomy/term/445/13/07b_UML-SysML-Overview.pdf

Learning Report - Embedded Software Design, Development Processes and Standard

GLOBAL ENGINEERING ACADEMY

Genesis

L&T Technology Services

Document History

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	Approved By	Remarks/Revision Details
1	10/10/2020	Vinay B J		Bhargav N	Added Activity-1, 2 & 3

Contents

Table of Figures

Table of Tables

Activity 1: White, Black, Grey box testing and its applications

White Box Testing

White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.

It is also called glass box testing or clear box testing or structural testing.

Working process of white box testing:

- Input: Requirements, Functional specifications, design documents, source code.
- Processing: Performing risk analysis for guiding through the entire process.
- Proper test planning: Designing test cases so as to cover entire code. Execute rinse-repeat until error-free software is reached. Also, the results are communicated.
- Output: Preparing final report of the entire testing process.

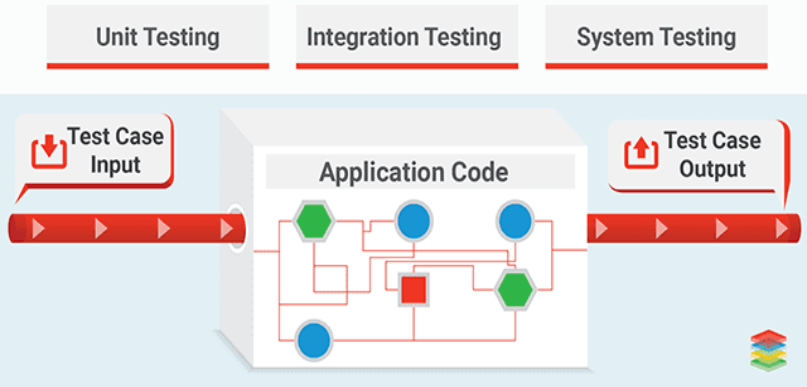

Figure 5: White Box Testing Approach

Reasons for white box testing:

- It identifies internal security holes.
- To check the way of input inside the code.
- Check the functionality of conditional loops.
- To test function, object, and statement at an individual level.

Types of White Box Testing:

- Path testing: In the path testing, we will write the flow graphs and test all independent paths.
- Loop testing: In the loop testing, we will test the loops such as while, for, and do-while, etc. and also check for ending condition if working correctly and if the size of the conditions is enough.
- Condition testing: In this, we will test all logical conditions for both true and false values; that is, we will verify for both if and else condition.

Advantages of White box testing:

- White box testing optimizes code so hidden errors can be identified.
- Test cases of white box testing can be easily automated.
- This testing is more thorough than other testing approaches as it covers all code paths.
- It can be started in the SDLC phase even without GUI.

Disadvantages of White box testing:

- White box testing is too much time consuming when it comes to large-scale programming applications.
- White box testing is much expensive and complex.
- It can lead to production error because it is not detailed by the developers.
- White box testing needs professional programmers who have a detailed knowledge and understanding of programming language and implementation.

Applications of White Box Testing:

- Security gaps and vulnerabilities - checking to see if security best practices were applied when coding the application, and if the code is vulnerable to known security threats and exploits.
- Broken or poorly structured paths - identifying conditional logic that is redundant, broken or inefficient.
- Expected output - executing all possible inputs to a function to see if it always returns the expected result.
- Loop testing - checking single loops, concatenated loops and nested loops for efficiency, conditional logic, and correct handling of local and global variables.
- Data Flow Testing (DFT) - tracking variables and their values as they pass through the code to find variables that are not correctly initialized, declared but never used, or incorrectly manipulated.

Black Box Testing

Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester.

Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

Working process of black box testing:

- Requirement: This is the initial stage of SDLC and in this stage, a requirement is gathered. Software testers also take part in this stage.
- Test Planning & Analysis: Testing Types applicable to the project are determined. A Test Plan is created which determines possible project risks and their mitigation.
- Design: In this stage Test cases/scripts are created on the basis of software requirement documents
- Test Execution: In this stage Test Cases prepared are executed. Bugs if any are fixed and re-tested.

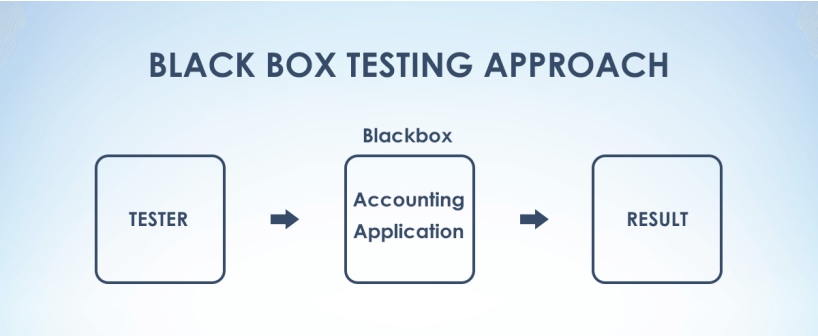

Figure 6: Black Box Testing Approach

Types of Black Box Testing:

- Functional testing: This black box testing type is related to the functional requirements of a system; it is done by software testers.
- Non-functional testing: This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.
- Regression testing: Regression Testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

Advantages of Black box testing:

- Well suited and efficient for large code segments.
- Code access is not required.
- Clearly separates user's perspective from the developer's perspective through visibly defined roles.
- Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language, or operating systems.

Disadvantages of Black box testing:

- Limited coverage, since only a selected number of test scenarios is actually performed.
- Inefficient testing, due to the fact that the tester only has limited knowledge about an application.
- Blind coverage, since the tester cannot target specific code segments or error-prone areas.
- The test cases are difficult to design.

Applications of Black Box Testing:

- Black box test evaluates all relevant subsystems, including UI/UX, web server or application server, database, dependencies, and integrated systems.
- Security technology that performs black box testing is Dynamic Application Security Testing (DAST), which tests products in staging or production and provides feedback on compliance and security issues.

Gray Box Testing

Gray Box Testing is a software testing technique which is a combination of Black Box Testing technique and White Box Testing technique. In Black Box Testing technique, tester is unknown to the internal structure of the item being tested and in White Box Testing the internal structure is known to tester. The internal structure is partially known in Gray Box Testing. This includes access to internal data structures and algorithms for purpose of designing the test cases.

Techniques used for Grey box Testing are:

- Matrix Testing: This testing technique involves defining all the variables that exist in their programs.
- Regression Testing**:** To check whether the change in the previous version has regressed other aspects of the program in the new version. It will be done by testing strategies like retest all, retest risky use cases, retest within a firewall.
- Orthogonal Array Testing or OAT**:** It provides maximum code coverage with minimum test cases.
- Pattern Testing: This testing is performed on the historical data of the previous system defects. Unlike black box testing, gray box testing digs within the code and determines why the failure happened.

Figure 7: Gray Box Testing Approach

Steps to perform Gray Box Testing

- Identify inputs
- Identify the outputs
- Identify the major paths
- Identify Sub functions
- Develop inputs for Sub functions
- Develop outputs for Sub functions
- Execute test case for Sub functions
- Verify the correct result for Sub functions
- Repeat steps 4 & 8 for other Sub functions
- Repeat steps 7 & 8 for other Sub functions

Advantages of Gray Box Testing:

- Users and developers have clear goals while doing testing.
- Gray box testing is mostly done by the user perspective.
- Testers are not required to have high programming skills for this testing.
- Gray box testing is non-intrusive.
- Overall quality of the product is improved.
- In Gray box testing, developers have more time for defect fixing.
- By doing Gray box testing, benefits of both black box and white box testing is obtained.
- Gray box testing is unbiased. It avoids conflicts between a tester and a developer.
- Gray box testing is much more effective in integration testing.

Disadvantages of Gray box testing:

- Defect association is difficult when Gray testing is performed for distributed systems.
- Limited access to internal structure leads to limited access for code path traversal.
- Because source code cannot be accessed, doing complete white box testing is not possible.
- Gray box testing is not suitable for algorithm testing.
- Most of the test cases are difficult to design.

Applications of Gray Box Testing:

- Grey-box testing is a perfect fit for Web-based applications.
- Grey-box testing is also a best approach for functional or domain testing.

Difference between Black Box, White Box and Gray Box Testing

Black Box Testing	White Box Testing	Gray Box Testing
The internal workings of an application need not be known.	The tester has limited knowledge of the internal workings of the application.	Tester has full knowledge of the internal workings of the application.
Also known as closed-box testing, data-driven testing, or functional testing.	Also known as translucent testing, as the tester has limited knowledge of the insides of the application.	Also known as clear-box testing, structural testing, or code-based testing.
Performed by end-users and also by testers and developers.	Performed by end-users and also by testers and developers.	Normally done by testers and developers.
Testing is based on external expectations - Internal behavior of the application is unknown.	Testing is done on the basis of high-level database diagrams and data flow diagrams.	Internal workings are fully known and the tester can design test data accordingly.
Not suited for algorithm testing.	Not suited for algorithm testing.	Suited for algorithm testing.
This can only be done by trial-and-error method.	Data domains and internal boundaries can be tested, if known.	Data domains and internal boundaries can be better tested.
It is the least time-consuming process among all the testing processes.	The entire testing process is the most time consuming among all the testing processes.	It is less time consuming than White Box testing.
It is less exhaustive than White Box and Grey Box testing methods.	It is most exhaustive than Black Box and Gray Box testing methods.	Partly exhaustive; depends upon the type of test cases are coding based or GUI based.
The base of this testing is external expectations internal behavior is unknown.	The base of this testing is coding which is responsible for internal working.	Testing based on high-level database diagrams and dataflow diagrams.

Table 2: Difference Between Black, White and Gray Box Testing

Activity 2: Various Standards and Tools

Aerospace Industry

AS9100 is the Quality Management System Requirements for Aerospace Manufacturers, but it also references several other standards that an organization must understand and integrate into their QMS.

Critical Standards for AS9100 Implementation:

- **ARP 9134A – Supply Chain Risk Management.** The guideline focuses on Quality as a key risk assessment factor taking into account elements from all aspects of the business having a direct link to global quality management.
- **AS5553C – Counterfeit Electronic Parts.** This standard is for use by organizations that procure and/or integrate EEE parts and/or assemblies containing such items.
- **AS9015A – Work Transfer Supplier Self Verification Process.** This standard defines the requirements for a delegation process identified in Aerospace Quality Management System standards.
- **AS9101 – QMS Audit Requirements**. This standard defines requirements for the preparation and execution of the audit process.
- **AS9102 – First Article Inspection.** This standard establishes the baseline requirements for performing and documenting FAI.
- **AS9103A – Key Characteristics**. This standard is primarily intended to apply to new parts and products, but can also be applied to parts currently in production.
- **AS9131C – Nonconformance.** This standard defines the common nonconformance data definition and documentation that must be exchanged between an internal/external supplier or sub-tier supplier, and the customer when informing about a nonconformity requiring formal decision.
- **AS9146 – Foreign Object Detection (FOD).** This standard defines FOD Prevention Program requirements for organizations that design, develop, and provide aviation, space, and defense products and services. And by organizations providing post-delivery support, including the provision of maintenance, spare parts, or materials for their own products and services.
- **ISO 10006 – Project Management.** This standard gives guidelines for the application of quality management in projects.
- **ISO 10007 – Configuration Management.** This standard provides guidance on the use of configuration management within an organization.
- **ISO 19011 – Guidelines for Auditing Management Systems.** This document provides guidance on auditing management systems.

AS9100 Core Tools.

- **ARP9013-1A –** Statistical Product Acceptance Requirements Using Isolated Lot Sampling Methods
- **ARP9013-2A –** Statistical Product Acceptance Requirements Using Attribute Or Variable Lot Acceptance Sampling Plans
- **ARP9013-3A –** Statistical Product Acceptance Requirements Using Process Control Methods

- **ARP9013-4A –** Statistical Product Acceptance Requirements Using Continuous Sampling, Skip-Lot Sampling, Or Methods for Special Cases
- **AS9145 –** Requirements for Advanced Product Quality Planning and Production Part Approval Process
- **ARP5580 –** Recommended Failure Modes and Effects Analysis (FMEA) Practices
- **ARP9136 –** Root Cause Analysis and Problem Solving
- **AS13004-** Process Failure Mode and Effects Analysis (PFMEA) and Control Plans

Aerospace Software

- **AS9115A** – Deliverable Aerospace Software Supplement for AS9100C (Supersedes AS9006:2003)
- **AS9006A** – Deliverable Aerospace Software Supplement for AS9100A
- **ARP9005B** – Aerospace Guidance for Non-Deliverable Software
- **ARP9034A** – A Process Standard for the Storage, Retrieval, and Use of 3D Type Design Data

Aerospace Supply Chain

- **ARP900B** – Aerospace Contract Clauses – used when subcontracting and purchasing from 3rd parties
- **ARP9107A** – Direct Delivery Authorization Guidance for Aerospace Companies
- **ARP9114A** – Direct Ship Guidance for Aerospace Companies
- **ARP9134A** – Supply Chain Risk Management Guidelines

Railway Industry

The following codes, standards and specifications applies to all systems and equipment, as the case maybe, forming part of the project:

- **NFPA 130:** Standard for Fixed Guideway Transit and Passenger Rail Systems: This standard specifies fire protection and life safety requirements for underground, surface, and elevated fixed guideway transit and passenger rail systems.
- **BS EN 50126:** Railway applications-The specification demonstrate reliability, availability, maintainability and safety (RAMS). This is a multi-part standards divided into many parts such as: o BS EN 50126-1: Railway applications. The specification and demonstration of reliability, availability, maintainability and safety (RAMS). Basic requirements and generic process
- **EN 50119** Railway applications - Fixed installations - Electric traction overhead contact lines
- **EN 50121**: Railway applications- Electromagnetic compatibility: This is a multi-part document divided into the following parts: o EN 50121-1: Railway applications. Electromagnetic compatibility. General
- **EN 50121-2:** Railway applications. Electromagnetic compatibility. Emission of the whole railway system to the outside world o EN 50121-3-1: Railway applications. Electromagnetic compatibility. Rolling stock. Train and complete vehicle
- **EN 50121-3-2:** Railway applications. Electromagnetic compatibility. Rolling stock. Apparatus o EN 50121-4: Railway applications. Electromagnetic compatibility. Emission and immunity of the signalling and telecommunications apparatus

- **EN 50121-5**: Railway applications. Electromagnetic compatibility. Emission and immunity of fixed power supply installations and apparatus.
- **BS EN 50122**: Railway applications, Fixed installations, Electrical safety, earthling and the return circuit: This is a multi-part document divided into the following parts:
- **BS EN 50122-1** Railway applications. Fixed installations. Protective provisions relating to electrical safety and earthing
- **BS EN 50122-2** Railway applications. Fixed installations. Protective provisions against the effects of stray currents caused by d.c. traction systems o BS EN 50122-3 Railway applications. Fixed installations. Electrical safety, earthing and the return circuit. Mutual Interaction of a.c. and d.c. traction systems
- **EN 50163:** RAILWAY APPLICATIONS - SUPPLY VOLTAGES OF TRACTION SYSTEMS: This European Standard specifies the main characteristics of the supply voltages of traction systems, such as traction fixed installations, including auxiliary devices fed by the contact line, and rolling stock, for use in the applications such as Railways, Guided mass transport systems such as tramways, elevated and underground railways mountain railways, and trolleybus systems and Material transportation systems.
- **IEC 60364 (4-41):** Electric installation of Buildings - Electric Shocks having following as applicable releases. o IEC 60364-4-41 Amd.1 Ed. 5.0 b:2017 (Amendment 1 - Low voltage electrical installations - Part 4-41: Protection for safety - Protection against electric shock) o IEC 60364-4-41 Ed. 5.0 b:2005 o IEC 60364-4-41 Ed. 5.1 b:2017
- **IEEE 80-2013:** Guide for Safety in AC Substation Grounding: This guide is primarily concerned with outdoor ac substations, either conventional or gas-insulated. These include distribution, transmission, and generating plant substations.
- **IEEE 519-2014:** Recommended Practice and Requirements for Harmonic Control in Electric Power Systems: Goals for the design of electrical systems that include both linear and nonlinear loads are established in this recommended practice.
- **IS 1893:** CRITERIA FOR EARTHQUAKE RESISTANT DESIGN OF STRUCTURES

Automobile Industry

- **ISO 14001**: As the international standard for environmental management systems — or EMS — ISO 14001 is the primary EMS certification for more than 250,000 organizations around the world. As the global standard for any business that wants to manage and positively control all aspects of its environmental impact, ISO 14001 certification is a proven way to demonstrate that you're serious about your business's environmental and economic sustainability.
- **ISO 45001:** Along with offering reliable products and services, automotive manufacturers must constantly strive to provide their workers and visitors with a safe and healthy business environment. With the ultimate goal of providing businesses with a framework for controlling and eliminating factors that can lead to illness, injuries and — in worst-case scenarios — death, obtaining ISO 45001 Health and Safety certification is a prudent move for any organization's senior management to support.

- **IATF 16949:** We work with the automotive industry to support the manufacturing of safe and reliable products, which are produced and continually improved to meet or exceed customer and regulatory authority requirements. Most organizations manufacturing for the automotive industry are required to be certificated to IATF 16949, which was developed by the International Automotive Task Force (IATF).
- **ISO 9001**: Since IATF 16949 isn't designed to be a self-sufficient quality standard but instead works best in conjunction with a comprehensive QMS, ISO 9001 certification makes a great deal of sense for automotive companies looking to demonstrate improvement in customer satisfaction, operating costs, stakeholder relationships, legal compliance, risk management, business credentials and attracting new business.
- **AIS-100:** Pedestrian protection
- **AIS-99:** Side mobile deformable offset
- **AIS-098:** Offset frontal crash
- **AUTOSAR:** It pursues the objective to create and establish an open and standardized software architecture for automotive electronic control units (ECUs).
- **ASIL-D:** An Automotive Safety Integrity Level (ASIL)-decomposed (or SIL-synthesized) architecture offers a reliable and robust path to achieving the highest levels of diagnostic coverage and gives end-equipment designers increased flexibility when developing safety-critical systems with high ASIL or SIL requirements.
- **ASPICE:** The Automotive SPICE Process Assessment Model (PAM) has been developed by consensus of the car manufacturers within the Automotive Special Interest Group (SIG) of the joint Procurement Forum / SPICE User Group under the Automotive SPICE initiative.
- **ISO26262:** It defines functional safety for automotive equipment applicable throughout the lifecycle of all automotive electronic and electrical safety-related systems.

Activity 3: Overview of diagrams used in SysML

- A graphical modelling language developed in response to the UML for Systems Engineering RFP developed by the OMG, INCOSE, and AP233a
- Supports the specification, analysis, design, verification, and validation of systems that include hardware, software, data, personnel, procedures, and facilities
- It is a visual modeling language that provides – Semantics = meaning, connected to a metamodel (rules governing the creation and the structure of models) – Notation = representation of meaning, graphical or textual.

SysML vs UML
UML is a general-purpose graphical modeling language aimed at Software Engineers

- Object diagram,
- Deployment diagram,
- Component diagram,
- Communication diagram,
- Timing diagram and
- Interaction overview diagram

Diagrams from UML

- Class diagram (Block Definition Diagram - Class→ Block)
- Package diagram
- Composite Structure diagram (Internal Block Diagram)
- State Machine Diagram
- Activity Diagram
- Use Case Diagram
- Sequence Diagram

In addition, SysML adds some new diagrams and constructs

- Parametric diagram,
- Requirement diagram
- Flow ports,
- Flow specifications
- Item flows.
- Allocation

SysML Extensions

- Blocks
- Item flows
- Value properties
- Allocations
- Requirements
- Parametrics
- Continuous flows

Available Diagrams

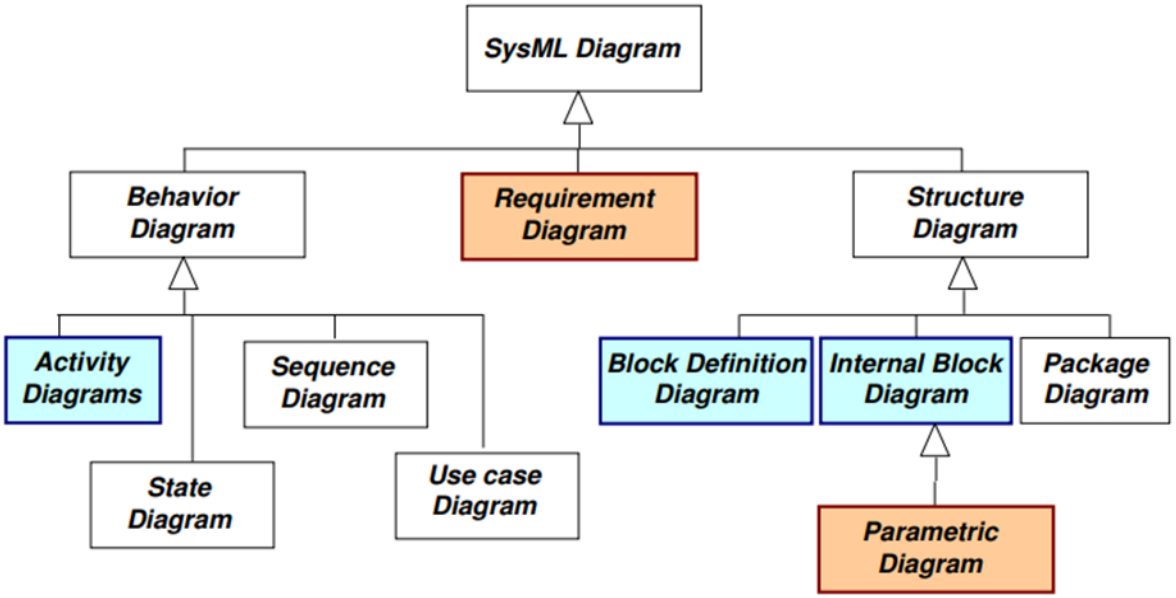

Figure 8: SysML Diagrams Overview

Behavior Diagram

The behavior diagrams include the use case diagram, activity diagram, sequence diagram, and state machine diagram.

Activity Diagram

The purpose of Activity diagrams is to specify dynamic system behaviors that Satisfy System Functional Requirements using both Control and Object (data) Flows. When properly applied Activity diagrams are recursively scalable. An Activity diagram shows system dynamic behavior using a combined Control Flow and Object (data) Flow model.

Sequence diagram

A Sequence diagram is a dynamic behavioral diagram that shows interactions (collaborations) among distributed objects or services via sequences of messages exchanged, along with corresponding (optional) events. The purpose of Sequence diagrams is to specify dynamic system behaviors as message-passing collaborations among prototypical Blocks (Parts). When properly applied (See Usage Notes below) Activity diagrams are recursively scalable and simulatable.

State Machine diagram

A State Machine diagram is a dynamic behavioral diagram that shows the sequences of States that an object or an interaction go through during its lifetime in response to Events (a.k.a. "Triggers"), which may result in side-effects (Actions). The purpose of State Machine diagrams is to specify dynamic system behaviors for time-critical, mission-critical, safety-critical, or financially-critical objects. When properly applied (See Usage Notes below) State Machine diagrams are recursively scalable and simulatable.

Use Case diagram

A Use Case diagram shows communications among system transactions (Use Cases) and external users (Actors) in the context of a system boundary (Subject; notation: rectangle). Actors may represent wetware (persons, organizations, facilities), software systems, or hardware systems. Defining relationships between the system Subject and the system Actors is an effective informal way to define system scope. The purpose of Use Case diagrams is to provide a high-level view of the subject system and convey the top-level system requirements in non-technical terms for all stakeholders, including customers and project managers as well as architects and engineers. Additional more rigorous SysML diagrams are needed to specify a scalable and simulatable System Architecture Model (SAM).

Requirement diagram

SysML predefines the following stereotype specializations of NFRs:

- Performance Requirement
- interface Requirement
- design Constraint
- physical Requirement

A SysML Requirement diagram is a static structural diagram that shows the relationships among Requirement constructs, model elements that Satisfy them, and Test Cases that Verify them.

The purpose of Requirement diagrams is to specify both Functional and Non-Functional Requirements within the model so that they can be traced to other model elements that Satisfy them and Test Cases that Verify them.

Structure Diagram

Block Definition Diagram

A Block Definition Diagram is a static structural diagram that shows system components, their contents (Properties, Behaviors, Constraints), Interfaces, and relationships.
Blocks can be recursively decomposed ("nested") into Parts by alternating between Block Definition Diagram (BDD) definitions and Internal Block Diagram (IBD) usages
Behaviors can either be encapsulated by Blocks (e.g., Operations, Signals, and State Machines) or Allocated (via «allocate» Dependency) to Blocks (e.g., Activities/Actions) directly or indirectly (via Interfaces).
Blocks can be mathematically constrained via Constraint Blocks to produce mathematically simulatable Parametric diagrams.
compare and contrast: UML 2 Class and Component diagrams; SA/SD System Context & Structure Chart diagrams; IDEF IDEF1X diagrams.

Purpose:

The purpose of Block Definition Diagrams is to specify system static structures that be used for Control Objects, Data Objects, and Interface Objects. When properly applied (See Usage Notes below) Block diagrams are recursively scalable and mathematically (parametrically) simulatable

Internal Block Diagram

An Internal Block Diagram is a static structural diagram owned by a particular Block that shows its encapsulated structural contents: Parts, Properties, Connectors, Ports, and Interfaces. Stated otherwise, an IBD is a "white-box" perspective of an encapsulated ("black-box") Block.
Blocks can be recursively decomposed ("nested") into Parts by alternating between Block Definition Diagram (BDD) definitions and Internal Block Diagram (IBD) usages
Behaviors can either be encapsulated by Blocks (e.g., Operations, Signals, and State Machines) or Allocated (via «allocate» Dependency) to Blocks (e.g., Activities/Actions) directly or indirectly (via Interfaces).

Blocks can be mathematically constrained via Constraint Blocks to produce mathematically simulatable Parametric diagrams.

compare and contrast: UML 2 Class and Component diagrams; SA/SD System Context & Structure Chart diagrams; IDEF IDEF1X diagrams.

Purpose:

The purpose of Internal Block Diagrams (IBDs) is to show the encapsulated structural contents (Parts, Properties, Connectors, Ports, Interfaces) of Blocks so that they can be recursively decomposed and "wired" using Interface Based Design techniques. When used correctly BDDs + IBDs are recursively scalable and mathematically (parametrically) simulatable

Parametric diagram

A Parametric diagram is a specialization of an Internal Block Diagram (IBD) that enforces mathematical rules (Constraints) defined by Constraint Blocks across the internal Part Value Properties bound by the Constraint Block Parameters.

Binding Connectors (keyword = «equal») between Constraint Block Parameters and internal Part Value Properties effect constraint satisfaction (propagation)

Purpose:

The purpose of Parametric diagrams (PARs) is to enforce mathematical rules across Block Value Properties. When used correctly BDDs + IBDs + PARs are recursively scalable and mathematically simulatable.

Package diagram

A Package diagram is a static structural diagram that shows the relationships among packages and their contents. Package can be stereotyped (customized) for organizing model elements into models, views, model libraries, and frameworks.

Purpose:

The purpose of Package diagram is to support the organization and management of large, complex System Architecture Models (SAMs).

References

- https://www.geeksforgeeks.org/software-engineering-white-box-testing/
- https://www.javatpoint.com/white-box-testing
- https://www.geeksforgeeks.org/software-engineering-black-box-testing/
- https://www.guru99.com/black-box-testing.html
- https://www.tutorialspoint.com/software_testing/software_testing_methods.htm
- https://www.geeksforgeeks.org/gray-box-testing-software-testing/
- https://www.guru99.com/grey-box-testing.html
- https://www.javatpoint.com/black-box-testing-vs-white-box-testing-vs-grey-box-testing
- https://www.tutorialspoint.com/software_testing/software_testing_methods.htm
- https://as9100store.com/aerospace-standards-explained/
- www.diva-portal.org/smash/get/diva2:430399/fulltext01.pdf
- inf.mit.bme.hu/sites/default/files/materials/taxonomy/term/445/13/07b_UML-SysML-Overview.pdf

Learning Report - Embedded Software Design, Development Processes and Standard

GLOBAL ENGINEERING ACADEMY

Genesis

L&T Technology Services

Document History

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	Approved By	Remarks/Revision Details
1	10/10/2020	Vinay B J		Bhargav N	Added Activity-1, 2 & 3

Contents

Table of Figures

Table of Tables

Activity 1: White, Black, Grey box testing and its applications

White Box Testing

White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.

It is also called glass box testing or clear box testing or structural testing.

Working process of white box testing:

- Input: Requirements, Functional specifications, design documents, source code.
- Processing: Performing risk analysis for guiding through the entire process.
- Proper test planning: Designing test cases so as to cover entire code. Execute rinse-repeat until error-free software is reached. Also, the results are communicated.
- Output: Preparing final report of the entire testing process.

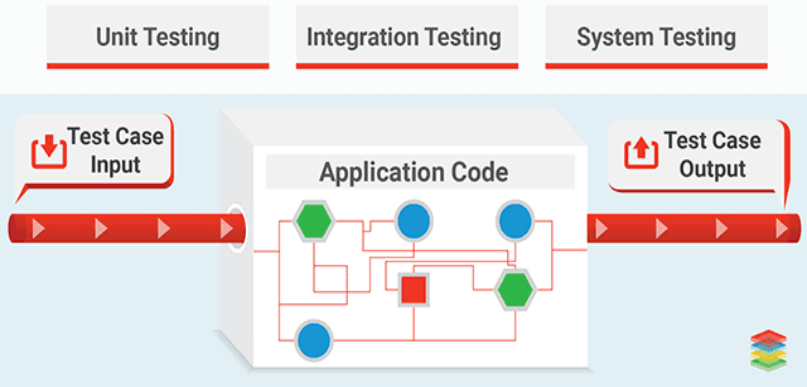

Figure 9: White Box Testing Approach

Reasons for white box testing:

- It identifies internal security holes.
- To check the way of input inside the code.
- Check the functionality of conditional loops.
- To test function, object, and statement at an individual level.

Types of White Box Testing:

- Path testing: In the path testing, we will write the flow graphs and test all independent paths.
- Loop testing: In the loop testing, we will test the loops such as while, for, and do-while, etc. and also check for ending condition if working correctly and if the size of the conditions is enough.
- Condition testing: In this, we will test all logical conditions for both true and false values; that is, we will verify for both if and else condition.

Advantages of White box testing:

- White box testing optimizes code so hidden errors can be identified.
- Test cases of white box testing can be easily automated.
- This testing is more thorough than other testing approaches as it covers all code paths.
- It can be started in the SDLC phase even without GUI.

Disadvantages of White box testing:

- White box testing is too much time consuming when it comes to large-scale programming applications.
- White box testing is much expensive and complex.
- It can lead to production error because it is not detailed by the developers.
- White box testing needs professional programmers who have a detailed knowledge and understanding of programming language and implementation.

Applications of White Box Testing:

- Security gaps and vulnerabilities - checking to see if security best practices were applied when coding the application, and if the code is vulnerable to known security threats and exploits.
- Broken or poorly structured paths - identifying conditional logic that is redundant, broken or inefficient.
- Expected output - executing all possible inputs to a function to see if it always returns the expected result.
- Loop testing - checking single loops, concatenated loops and nested loops for efficiency, conditional logic, and correct handling of local and global variables.
- Data Flow Testing (DFT) - tracking variables and their values as they pass through the code to find variables that are not correctly initialized, declared but never used, or incorrectly manipulated.

Black Box Testing

Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester.

Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

Working process of black box testing:

- Requirement: This is the initial stage of SDLC and in this stage, a requirement is gathered. Software testers also take part in this stage.
- Test Planning & Analysis: Testing Types applicable to the project are determined. A Test Plan is created which determines possible project risks and their mitigation.
- Design: In this stage Test cases/scripts are created on the basis of software requirement documents
- Test Execution: In this stage Test Cases prepared are executed. Bugs if any are fixed and re-tested.

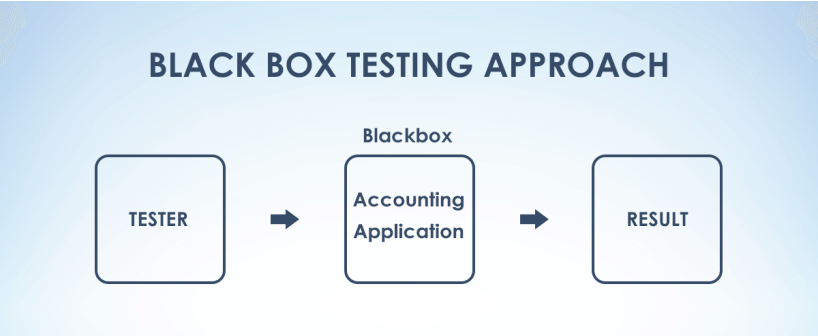

Figure 10: Black Box Testing Approach

Types of Black Box Testing:

- Functional testing: This black box testing type is related to the functional requirements of a system; it is done by software testers.
- Non-functional testing: This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.
- Regression testing: Regression Testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

Advantages of Black box testing:

- Well suited and efficient for large code segments.
- Code access is not required.
- Clearly separates user's perspective from the developer's perspective through visibly defined roles.
- Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language, or operating systems.

Disadvantages of Black box testing:

- Limited coverage, since only a selected number of test scenarios is actually performed.
- Inefficient testing, due to the fact that the tester only has limited knowledge about an application.
- Blind coverage, since the tester cannot target specific code segments or error-prone areas.
- The test cases are difficult to design.

Applications of Black Box Testing:

- Black box test evaluates all relevant subsystems, including UI/UX, web server or application server, database, dependencies, and integrated systems.
- Security technology that performs black box testing is Dynamic Application Security Testing (DAST), which tests products in staging or production and provides feedback on compliance and security issues.

Gray Box Testing

Gray Box Testing is a software testing technique which is a combination of Black Box Testing technique and White Box Testing technique. In Black Box Testing technique, tester is unknown to the internal structure of the item being tested and in White Box Testing the internal structure is known to tester. The internal structure is partially known in Gray Box Testing. This includes access to internal data structures and algorithms for purpose of designing the test cases.

Techniques used for Grey box Testing are:

- Matrix Testing: This testing technique involves defining all the variables that exist in their programs.
- Regression Testing**:** To check whether the change in the previous version has regressed other aspects of the program in the new version. It will be done by testing strategies like retest all, retest risky use cases, retest within a firewall.
- Orthogonal Array Testing or OAT**:** It provides maximum code coverage with minimum test cases.
- Pattern Testing: This testing is performed on the historical data of the previous system defects. Unlike black box testing, gray box testing digs within the code and determines why the failure happened.

Figure 11: Gray Box Testing Approach

Steps to perform Gray Box Testing

- Identify inputs
- Identify the outputs
- Identify the major paths
- Identify Sub functions
- Develop inputs for Sub functions
- Develop outputs for Sub functions
- Execute test case for Sub functions
- Verify the correct result for Sub functions
- Repeat steps 4 & 8 for other Sub functions
- Repeat steps 7 & 8 for other Sub functions

Advantages of Gray Box Testing:

- Users and developers have clear goals while doing testing.
- Gray box testing is mostly done by the user perspective.
- Testers are not required to have high programming skills for this testing.
- Gray box testing is non-intrusive.
- Overall quality of the product is improved.
- In Gray box testing, developers have more time for defect fixing.
- By doing Gray box testing, benefits of both black box and white box testing is obtained.
- Gray box testing is unbiased. It avoids conflicts between a tester and a developer.
- Gray box testing is much more effective in integration testing.

Disadvantages of Gray box testing:

- Defect association is difficult when Gray testing is performed for distributed systems.
- Limited access to internal structure leads to limited access for code path traversal.
- Because source code cannot be accessed, doing complete white box testing is not possible.
- Gray box testing is not suitable for algorithm testing.
- Most of the test cases are difficult to design.

Applications of Gray Box Testing:

- Grey-box testing is a perfect fit for Web-based applications.
- Grey-box testing is also a best approach for functional or domain testing.

Difference between Black Box, White Box and Gray Box Testing

Black Box Testing	White Box Testing	Gray Box Testing
The internal workings of an application need not be known.	The tester has limited knowledge of the internal workings of the application.	Tester has full knowledge of the internal workings of the application.
Also known as closed-box testing, data-driven testing, or functional testing.	Also known as translucent testing, as the tester has limited knowledge of the insides of the application.	Also known as clear-box testing, structural testing, or code-based testing.
Performed by end-users and also by testers and developers.	Performed by end-users and also by testers and developers.	Normally done by testers and developers.
Testing is based on external expectations - Internal behavior of the application is unknown.	Testing is done on the basis of high-level database diagrams and data flow diagrams.	Internal workings are fully known and the tester can design test data accordingly.
Not suited for algorithm testing.	Not suited for algorithm testing.	Suited for algorithm testing.
This can only be done by trial-and-error method.	Data domains and internal boundaries can be tested, if known.	Data domains and internal boundaries can be better tested.
It is the least time-consuming process among all the testing processes.	The entire testing process is the most time consuming among all the testing processes.	It is less time consuming than White Box testing.
It is less exhaustive than White Box and Grey Box testing methods.	It is most exhaustive than Black Box and Gray Box testing methods.	Partly exhaustive; depends upon the type of test cases are coding based or GUI based.
The base of this testing is external expectations internal behavior is unknown.	The base of this testing is coding which is responsible for internal working.	Testing based on high-level database diagrams and dataflow diagrams.

Table 3: Difference Between Black, White and Gray Box Testing

Activity 2: Various Standards and Tools

Aerospace Industry

AS9100 is the Quality Management System Requirements for Aerospace Manufacturers, but it also references several other standards that an organization must understand and integrate into their QMS.

Critical Standards for AS9100 Implementation:

- **ARP 9134A – Supply Chain Risk Management.** The guideline focuses on Quality as a key risk assessment factor taking into account elements from all aspects of the business having a direct link to global quality management.
- **AS5553C – Counterfeit Electronic Parts.** This standard is for use by organizations that procure and/or integrate EEE parts and/or assemblies containing such items.
- **AS9015A – Work Transfer Supplier Self Verification Process.** This standard defines the requirements for a delegation process identified in Aerospace Quality Management System standards.
- **AS9101 – QMS Audit Requirements**. This standard defines requirements for the preparation and execution of the audit process.
- **AS9102 – First Article Inspection.** This standard establishes the baseline requirements for performing and documenting FAI.
- **AS9103A – Key Characteristics**. This standard is primarily intended to apply to new parts and products, but can also be applied to parts currently in production.
- **AS9131C – Nonconformance.** This standard defines the common nonconformance data definition and documentation that must be exchanged between an internal/external supplier or sub-tier supplier, and the customer when informing about a nonconformity requiring formal decision.
- **AS9146 – Foreign Object Detection (FOD).** This standard defines FOD Prevention Program requirements for organizations that design, develop, and provide aviation, space, and defense products and services. And by organizations providing post-delivery support, including the provision of maintenance, spare parts, or materials for their own products and services.
- **ISO 10006 – Project Management.** This standard gives guidelines for the application of quality management in projects.
- **ISO 10007 – Configuration Management.** This standard provides guidance on the use of configuration management within an organization.
- **ISO 19011 – Guidelines for Auditing Management Systems.** This document provides guidance on auditing management systems.

AS9100 Core Tools.

- **ARP9013-1A –** Statistical Product Acceptance Requirements Using Isolated Lot Sampling Methods
- **ARP9013-2A –** Statistical Product Acceptance Requirements Using Attribute Or Variable Lot Acceptance Sampling Plans
- **ARP9013-3A –** Statistical Product Acceptance Requirements Using Process Control Methods

- **ARP9013-4A –** Statistical Product Acceptance Requirements Using Continuous Sampling, Skip-Lot Sampling, Or Methods for Special Cases
- **AS9145 –** Requirements for Advanced Product Quality Planning and Production Part Approval Process
- **ARP5580 –** Recommended Failure Modes and Effects Analysis (FMEA) Practices
- **ARP9136 –** Root Cause Analysis and Problem Solving
- **AS13004-** Process Failure Mode and Effects Analysis (PFMEA) and Control Plans

Aerospace Software

- **AS9115A** – Deliverable Aerospace Software Supplement for AS9100C (Supersedes AS9006:2003)
- **AS9006A** – Deliverable Aerospace Software Supplement for AS9100A
- **ARP9005B** – Aerospace Guidance for Non-Deliverable Software
- **ARP9034A** – A Process Standard for the Storage, Retrieval, and Use of 3D Type Design Data

Aerospace Supply Chain

- **ARP900B** – Aerospace Contract Clauses – used when subcontracting and purchasing from 3rd parties
- **ARP9107A** – Direct Delivery Authorization Guidance for Aerospace Companies
- **ARP9114A** – Direct Ship Guidance for Aerospace Companies
- **ARP9134A** – Supply Chain Risk Management Guidelines

Railway Industry

The following codes, standards and specifications applies to all systems and equipment, as the case maybe, forming part of the project:

- **NFPA 130:** Standard for Fixed Guideway Transit and Passenger Rail Systems: This standard specifies fire protection and life safety requirements for underground, surface, and elevated fixed guideway transit and passenger rail systems.
- **BS EN 50126:** Railway applications-The specification demonstrate reliability, availability, maintainability and safety (RAMS). This is a multi-part standards divided into many parts such as: o BS EN 50126-1: Railway applications. The specification and demonstration of reliability, availability, maintainability and safety (RAMS). Basic requirements and generic process
- **EN 50119** Railway applications - Fixed installations - Electric traction overhead contact lines
- **EN 50121**: Railway applications- Electromagnetic compatibility: This is a multi-part document divided into the following parts: o EN 50121-1: Railway applications. Electromagnetic compatibility. General
- **EN 50121-2:** Railway applications. Electromagnetic compatibility. Emission of the whole railway system to the outside world o EN 50121-3-1: Railway applications. Electromagnetic compatibility. Rolling stock. Train and complete vehicle
- **EN 50121-3-2:** Railway applications. Electromagnetic compatibility. Rolling stock. Apparatus o EN 50121-4: Railway applications. Electromagnetic compatibility. Emission and immunity of the signalling and telecommunications apparatus

- **EN 50121-5**: Railway applications. Electromagnetic compatibility. Emission and immunity of fixed power supply installations and apparatus.
- **BS EN 50122**: Railway applications, Fixed installations, Electrical safety, earthling and the return circuit: This is a multi-part document divided into the following parts:
- **BS EN 50122-1** Railway applications. Fixed installations. Protective provisions relating to electrical safety and earthing
- **BS EN 50122-2** Railway applications. Fixed installations. Protective provisions against the effects of stray currents caused by d.c. traction systems o BS EN 50122-3 Railway applications. Fixed installations. Electrical safety, earthing and the return circuit. Mutual Interaction of a.c. and d.c. traction systems
- **EN 50163:** RAILWAY APPLICATIONS - SUPPLY VOLTAGES OF TRACTION SYSTEMS: This European Standard specifies the main characteristics of the supply voltages of traction systems, such as traction fixed installations, including auxiliary devices fed by the contact line, and rolling stock, for use in the applications such as Railways, Guided mass transport systems such as tramways, elevated and underground railways mountain railways, and trolleybus systems and Material transportation systems.
- **IEC 60364 (4-41):** Electric installation of Buildings - Electric Shocks having following as applicable releases. o IEC 60364-4-41 Amd.1 Ed. 5.0 b:2017 (Amendment 1 - Low voltage electrical installations - Part 4-41: Protection for safety - Protection against electric shock) o IEC 60364-4-41 Ed. 5.0 b:2005 o IEC 60364-4-41 Ed. 5.1 b:2017
- **IEEE 80-2013:** Guide for Safety in AC Substation Grounding: This guide is primarily concerned with outdoor ac substations, either conventional or gas-insulated. These include distribution, transmission, and generating plant substations.
- **IEEE 519-2014:** Recommended Practice and Requirements for Harmonic Control in Electric Power Systems: Goals for the design of electrical systems that include both linear and nonlinear loads are established in this recommended practice.
- **IS 1893:** CRITERIA FOR EARTHQUAKE RESISTANT DESIGN OF STRUCTURES

Automobile Industry

- **ISO 14001**: As the international standard for environmental management systems — or EMS — ISO 14001 is the primary EMS certification for more than 250,000 organizations around the world. As the global standard for any business that wants to manage and positively control all aspects of its environmental impact, ISO 14001 certification is a proven way to demonstrate that you're serious about your business's environmental and economic sustainability.
- **ISO 45001:** Along with offering reliable products and services, automotive manufacturers must constantly strive to provide their workers and visitors with a safe and healthy business environment. With the ultimate goal of providing businesses with a framework for controlling and eliminating factors that can lead to illness, injuries and — in worst-case scenarios — death, obtaining ISO 45001 Health and Safety certification is a prudent move for any organization's senior management to support.

- **IATF 16949:** We work with the automotive industry to support the manufacturing of safe and reliable products, which are produced and continually improved to meet or exceed customer and regulatory authority requirements. Most organizations manufacturing for the automotive industry are required to be certificated to IATF 16949, which was developed by the International Automotive Task Force (IATF).
- **ISO 9001**: Since IATF 16949 isn't designed to be a self-sufficient quality standard but instead works best in conjunction with a comprehensive QMS, ISO 9001 certification makes a great deal of sense for automotive companies looking to demonstrate improvement in customer satisfaction, operating costs, stakeholder relationships, legal compliance, risk management, business credentials and attracting new business.
- **AIS-100:** Pedestrian protection
- **AIS-99:** Side mobile deformable offset
- **AIS-098:** Offset frontal crash
- **AUTOSAR:** It pursues the objective to create and establish an open and standardized software architecture for automotive electronic control units (ECUs).
- **ASIL-D:** An Automotive Safety Integrity Level (ASIL)-decomposed (or SIL-synthesized) architecture offers a reliable and robust path to achieving the highest levels of diagnostic coverage and gives end-equipment designers increased flexibility when developing safety-critical systems with high ASIL or SIL requirements.
- **ASPICE:** The Automotive SPICE Process Assessment Model (PAM) has been developed by consensus of the car manufacturers within the Automotive Special Interest Group (SIG) of the joint Procurement Forum / SPICE User Group under the Automotive SPICE initiative.
- **ISO26262:** It defines functional safety for automotive equipment applicable throughout the lifecycle of all automotive electronic and electrical safety-related systems.

Activity 3: Overview of diagrams used in SysML

- A graphical modelling language developed in response to the UML for Systems Engineering RFP developed by the OMG, INCOSE, and AP233a
- Supports the specification, analysis, design, verification, and validation of systems that include hardware, software, data, personnel, procedures, and facilities
- It is a visual modeling language that provides – Semantics = meaning, connected to a metamodel (rules governing the creation and the structure of models) – Notation = representation of meaning, graphical or textual.

SysML vs UML
UML is a general-purpose graphical modeling language aimed at Software Engineers

- Object diagram,
- Deployment diagram,
- Component diagram,
- Communication diagram,
- Timing diagram and
- Interaction overview diagram

Diagrams from UML

- Class diagram (Block Definition Diagram - Class→ Block)
- Package diagram
- Composite Structure diagram (Internal Block Diagram)
- State Machine Diagram
- Activity Diagram
- Use Case Diagram
- Sequence Diagram

In addition, SysML adds some new diagrams and constructs

- Parametric diagram,
- Requirement diagram
- Flow ports,
- Flow specifications
- Item flows.
- Allocation

SysML Extensions

- Blocks
- Item flows
- Value properties
- Allocations
- Requirements
- Parametrics
- Continuous flows

Available Diagrams

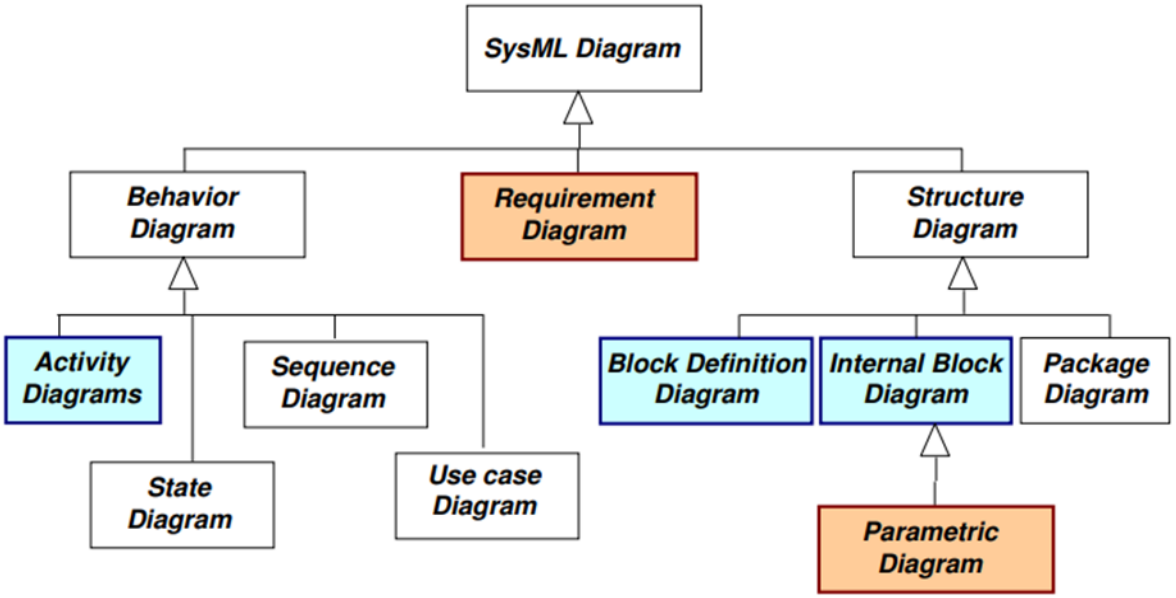

Figure 12: SysML Diagrams Overview

Behavior Diagram

The behavior diagrams include the use case diagram, activity diagram, sequence diagram, and state machine diagram.

Activity Diagram

The purpose of Activity diagrams is to specify dynamic system behaviors that Satisfy System Functional Requirements using both Control and Object (data) Flows. When properly applied Activity diagrams are recursively scalable. An Activity diagram shows system dynamic behavior using a combined Control Flow and Object (data) Flow model.

Sequence diagram

A Sequence diagram is a dynamic behavioral diagram that shows interactions (collaborations) among distributed objects or services via sequences of messages exchanged, along with corresponding (optional) events. The purpose of Sequence diagrams is to specify dynamic system behaviors as message-passing collaborations among prototypical Blocks (Parts). When properly applied (See Usage Notes below) Activity diagrams are recursively scalable and simulatable.

State Machine diagram

A State Machine diagram is a dynamic behavioral diagram that shows the sequences of States that an object or an interaction go through during its lifetime in response to Events (a.k.a. "Triggers"), which may result in side-effects (Actions). The purpose of State Machine diagrams is to specify dynamic system behaviors for time-critical, mission-critical, safety-critical, or financially-critical objects. When properly applied (See Usage Notes below) State Machine diagrams are recursively scalable and simulatable.

Use Case diagram

A Use Case diagram shows communications among system transactions (Use Cases) and external users (Actors) in the context of a system boundary (Subject; notation: rectangle). Actors may represent wetware (persons, organizations, facilities), software systems, or hardware systems. Defining relationships between the system Subject and the system Actors is an effective informal way to define system scope. The purpose of Use Case diagrams is to provide a high-level view of the subject system and convey the top-level system requirements in non-technical terms for all stakeholders, including customers and project managers as well as architects and engineers. Additional more rigorous SysML diagrams are needed to specify a scalable and simulatable System Architecture Model (SAM).

Requirement diagram

SysML predefines the following stereotype specializations of NFRs:

- Performance Requirement
- interface Requirement
- design Constraint
- physical Requirement

A SysML Requirement diagram is a static structural diagram that shows the relationships among Requirement constructs, model elements that Satisfy them, and Test Cases that Verify them.

The purpose of Requirement diagrams is to specify both Functional and Non-Functional Requirements within the model so that they can be traced to other model elements that Satisfy them and Test Cases that Verify them.

Structure Diagram

Block Definition Diagram

A Block Definition Diagram is a static structural diagram that shows system components, their contents (Properties, Behaviors, Constraints), Interfaces, and relationships.
Blocks can be recursively decomposed ("nested") into Parts by alternating between Block Definition Diagram (BDD) definitions and Internal Block Diagram (IBD) usages
Behaviors can either be encapsulated by Blocks (e.g., Operations, Signals, and State Machines) or Allocated (via «allocate» Dependency) to Blocks (e.g., Activities/Actions) directly or indirectly (via Interfaces).
Blocks can be mathematically constrained via Constraint Blocks to produce mathematically simulatable Parametric diagrams.
compare and contrast: UML 2 Class and Component diagrams; SA/SD System Context & Structure Chart diagrams; IDEF IDEF1X diagrams.

Purpose:

The purpose of Block Definition Diagrams is to specify system static structures that be used for Control Objects, Data Objects, and Interface Objects. When properly applied (See Usage Notes below) Block diagrams are recursively scalable and mathematically (parametrically) simulatable

Internal Block Diagram

An Internal Block Diagram is a static structural diagram owned by a particular Block that shows its encapsulated structural contents: Parts, Properties, Connectors, Ports, and Interfaces. Stated otherwise, an IBD is a "white-box" perspective of an encapsulated ("black-box") Block.
Blocks can be recursively decomposed ("nested") into Parts by alternating between Block Definition Diagram (BDD) definitions and Internal Block Diagram (IBD) usages
Behaviors can either be encapsulated by Blocks (e.g., Operations, Signals, and State Machines) or Allocated (via «allocate» Dependency) to Blocks (e.g., Activities/Actions) directly or indirectly (via Interfaces).

Blocks can be mathematically constrained via Constraint Blocks to produce mathematically simulatable Parametric diagrams.

compare and contrast: UML 2 Class and Component diagrams; SA/SD System Context & Structure Chart diagrams; IDEF IDEF1X diagrams.

Purpose:

The purpose of Internal Block Diagrams (IBDs) is to show the encapsulated structural contents (Parts, Properties, Connectors, Ports, Interfaces) of Blocks so that they can be recursively decomposed and "wired" using Interface Based Design techniques. When used correctly BDDs + IBDs are recursively scalable and mathematically (parametrically) simulatable

Parametric diagram

A Parametric diagram is a specialization of an Internal Block Diagram (IBD) that enforces mathematical rules (Constraints) defined by Constraint Blocks across the internal Part Value Properties bound by the Constraint Block Parameters.

Binding Connectors (keyword = «equal») between Constraint Block Parameters and internal Part Value Properties effect constraint satisfaction (propagation)

Purpose:

The purpose of Parametric diagrams (PARs) is to enforce mathematical rules across Block Value Properties. When used correctly BDDs + IBDs + PARs are recursively scalable and mathematically simulatable.

Package diagram

A Package diagram is a static structural diagram that shows the relationships among packages and their contents. Package can be stereotyped (customized) for organizing model elements into models, views, model libraries, and frameworks.

Purpose:

The purpose of Package diagram is to support the organization and management of large, complex System Architecture Models (SAMs).

References

- https://www.geeksforgeeks.org/software-engineering-white-box-testing/
- https://www.javatpoint.com/white-box-testing
- https://www.geeksforgeeks.org/software-engineering-black-box-testing/
- https://www.guru99.com/black-box-testing.html
- https://www.tutorialspoint.com/software_testing/software_testing_methods.htm
- https://www.geeksforgeeks.org/gray-box-testing-software-testing/
- https://www.guru99.com/grey-box-testing.html
- https://www.javatpoint.com/black-box-testing-vs-white-box-testing-vs-grey-box-testing
- https://www.tutorialspoint.com/software_testing/software_testing_methods.htm
- https://as9100store.com/aerospace-standards-explained/
- www.diva-portal.org/smash/get/diva2:430399/fulltext01.pdf
- inf.mit.bme.hu/sites/default/files/materials/taxonomy/term/445/13/07b_UML-SysML-Overview.pdf