An optimized process for developing cyber-physical product systems (CPS) through the integration of model-based development (MBSE), implemented based on the V-model and a team topology approach.

Dipl.- Ing. Stefan Peil

Zur Eiche 14

Weimar (Lahn) 35096 Germany

✉ stefan.peil@stpse.de

Summary

The development of safety-critical cyber-physical (product) systems (CPS) faces challenges due to inefficient team structures and fragmented processes often caused by the traditional V-model. Although this model is proven and recognized as a standard in developing cyber-physical systems, combined with conventional silo-like team approaches, it often leads to communication barriers, rework, and incorrect software and hardware components integration.

Team topologies, a modern approach to team organization, in conjunction with Model-Based Systems Engineering (MBSE), offer a forward-looking solution.

Combining these methods promotes collaboration between development and validation teams and ensures consistency throughout the development cycle and traceability of requirements.

The stream-aligned, enabler, platform, and core teams created with the team topology approach are strategically aligned with the phases of the V-model to overcome development silos and ensure holistic system responsibility.

Pre-Release: January 13, 2025

The current situation in project- and product design in the context of the development of cyber-physical systems (CPS)

Challenges in the traditional implementation of the V-model in systems- and product development

The V-model is a fundamental framework for developing networked systems, especially cyber-physical systems that must meet high-security standards, e.g., the CRA, the Cyber Resilience Act of the European Union, or the regulatory requirements of medical technology. The V-model is characterized by its structured and logical approach, which generically outlines the individual phases for CPS design and validation and thus enables effective development and validation processes.

The V-model is used so that the two branches of the V are divided into development, verification, and validation phases. The left branch of the V represents the actual development with its specific phases, and the verification and validation phases are assigned to the right branch of the V.

However, one of the author's observations is that the assignment of team roles along the left-hand side of the V-model (which represents the development phases) and the right-hand side (which represents the verification and validation phases) are primarily done separately, which often leads to problems.

The problems arise from a lack of consistency and coordination of the development activities and, thus, incorrect integration of the development results (artifacts) of the different development domains (software and hardware development) with regard to the planned development phases. This is caused by unfavorable team communication and cooperation and worsens the overall efficiency due to the re-development and reworking of the project.

The V-model: Overview

The V-model divides the development process into successive phases, each accompanied by a corresponding test phase, up to the final validation systemically from top to bottom or from a high level of abstraction to the concrete design. It emphasizes the close relationship between each development phase and the corresponding test and ensures that each phase of the system is traceably tested for its requirements regarding its artifacts.

Explanations of the V-model

The "V" shape stands for:

- **Left branch (development)**: The top-down development phases that lead from the abstract system concept to concrete implementation.
- **Right branch (test/verification/validation)**: The appropriate validation and testing phases that ensure that what has been developed meets the intended goals
- **Symmetry:** The V-model is designed to mirror a test phase at the same level on the right-hand branch at each development phase. This graphically shows the relationship between the development stage's level of abstraction and a corresponding verification and validation at the same level of abstraction.

Phases of the V-model

Left branch:
(development phases - from top to bottom / from high abstraction to the concrete)

1. **System concept definition (or operational concept)**:
    - High-level understanding of what the system needs to achieve.

- Focus on the stakeholders' objectives, the context of the technological system, and the business case.

1. **System requirement determination**:
   - Detailed abstract specification of what the overall system must achieve.
   - Contains functional and non-functional requirements.
   - Defines the development limitations (constraints).
2. **Functional architecture definition**:
   - Breakdown of the system into its core functions.
   - Design of the high-level information flow or behavior and its interface definitions (APIs, Application Programming Interface) in a structural representation (architecture)
3. **Logical architecture definition**:
   - Conversion of functional system requirements and functions into logical system components without specifying concrete development solutions
   - Defines the interactions between software and hardware

   Note: This is the phase in development in which genuine innovations can be generated.
4. **Cyber-physical concrete architecture design**:
   - Detailed design of the system's physical (hardware) and cyber (software) components.
   - Contains hardware schematics and software architecture.
   - Describes in detail the structure and behavior of concrete solutions
5. **Specifications for software and hardware components based on the architecture**:
   - Final, detailed instructions for implementing the system design as specifications in software and hardware areas.
6. **Implementation**:
   - Coding, structure, or composition of the product or system components.

Right branch (validation phases from bottom to top/bottom-up)

1. **Component test**:
   - Check whether individual components function as specified.
2. **Integration test**:
   - Ensures that individual concrete development components are functional and work correctly with the other components. Interfaces are checked, and the correct functioning of the individual components. This often takes place on test benches and not yet in the actual operating environment
3. **System composition**:
   - All components are assembled into the final system in the operating environment.

Dipl.- Ing. Stefan Peil | Zur Eiche 14 | 35096 Weimar (Lahn) | Germany

4. **System validation**:
    - Validates the entire system based on the original concept and system requirements.
    - Includes field testing and stakeholder feedback.
    - Includes tests within the specified operating framework (system specification) and tests at the system specification's limits (stress testst) with all possible operating variables analyzed during development.

Basic principles of the V-model

1. **Verification:**
    - Ensures that "we build the system correctly".
    - Focuses on confirming that each development phase is implemented correctly.
2. **Validation:**
    - Ensures that "we build the right system".
    - It focuses on meeting the needs of stakeholders, the original system objectives, and the operational concept.
3. **Consistency**
    - Ensures coherent and logical development, verification, and validation. Development phases and steps are carried out in a way that builds on each other and can be transferred and controlled in project management.
4. **Bidirectional traceability:**
    - Each phase on the left has a corresponding test phase on the right, ensuring a complete requirement check.

Advantages of the V-model

1. **Clarity and structure:**
    - Clearly defined phases make it easier to track progress, as each phase can be viewed as a milestone. The coherent structure can be used for good release management.
      The development versions of the individual domains (software and hardware) are integrated (configured) into an overall system version (release) and tested.
2. **Error avoidance**
    - Problems are identified before implementation through development reviews and phase approvals.
3. **Tests traceable to requirements:**
    - Stringent logic ensures traceable system validation, facilitating technical documentation and approval.

Identified limitations of the V-model in practice

1. **Lack of flexibility**
    - The teams expect the requirements to be defined and established early, which is a challenge in dynamic development projects and environments.

- Not ideal for highly recursive or explorative processes.
- Inadmissible simplification of the development approach, as the V-model can be misinterpreted as a waterfall model.
- The late discovery of integration problems due to inadmissible simplification can be costly.

2. **Silo-like team structures**

- Development teams concentrate in silos on assigned tasks (concepts, requirements, design, tests, left and right branches of the V-model).
- Test teams are isolated until the downstream phases (validation, integration) and must wait for the development results.
- Limited cross-functional collaboration leads to inconsistencies, lack of traceability, and inefficiencies.

3. **Fragmented communication**

- The lack of continuous interaction between design, implementation, and validation teams leads to incomplete and complex feedback loops.
- Problems identified during testing often require costly rework.

4. **Isolated areas**

- Cyber and physical areas are treated independently of each other, which makes holistic system integration difficult.
- Disjointed team structures mean that cross-departmental dependencies cannot be managed effectively.

5. **Complexity of project management**

- Project managers have difficulty coordinating tasks for the specialized teams, which leads to delays and misunderstandings.

Model-based systems engineering (MBSE): Overview

Model-Based Systems Engineering (MBSE) is an approach to systems engineering that uses cohesive digital, graphical models as the primary means of communication, analysis, and documentation throughout the system lifecycle. In contrast to traditional document-centered systems engineering (DBSE), MBSE replaces lengthy text specifications with unambiguous, interactive models that represent various system aspects such as requirements, design, implementation, and validation.

MBSE is particularly valuable in complex projects where different technical disciplines (e.g., software, hardware, actuators) need to work together to deliver integrated systems. By using standards-based (globally standardized) modeling languages such as SysML (Systems Modeling Language) or UML (Unified Modeling Language), MBSE ensures complete consistency, traceability, and a holistic system view. The author sees MBSE as the lingua franca of system developers, as the models can be understood cross-functionally and cross-culturally, which is not the case with DBSE.

Basic principles of MBSE

1. **Model-centered approach**:

- Models are the "primary source of truth" ("single source of truth") for system definitions, requirements, and designs.

- Stakeholders access these models in their work to ensure uniform alignment and understanding of the system.

2. **Interdisciplinary cooperation**

   - MBSE integrates perspectives from different areas (e.g., hardware, actuators, software) into a standardized model. The most significant advantage over DBSE is the generation of specific views for the stakeholders and development teams, as with an architectural draughtsman's work.

   - MBSE promotes communication and understanding between the teams.

3. **Life cycle integration:**

   - Models evolve throughout the system lifecycle and support activities from concept to validation.

   - Enables iterative updates and real-time feedback.

   - Models are a digital twin of a development process and the system to be developed.

   - Models, especially logical architectures, are designed in such a way that they can be reused for new projects.

4. **Traceability:**

   - MBSE ensures that requirements are linked to design decisions, tests, and system elements, thus enabling end-to-end traceability.

5. **Standards and tools:**

   - Frequently used standards include SysML, UML, and cross-domain tools such as SPARX Enterprise Architect.

Identified advantages of MBSE in practice

MBSE avoids challenges associated with traditional document-centric approaches (BDSE) by:

1. **Improving consistency**

   - Centralized models reduce the risk of contradictory documentation.

   Note: However, it is essential to ensure that the modeling follows objectives and agreements that are jointly agreed upon with the stakeholders. Modeling "at the drop of a hat" will lead to failure of the approach, as the overview is quickly lost, and it becomes impossible to maintain the model. Modeling for the sake of "modeling" must also be avoided.

2. **Improving traceability**

   - The relationships between requirements, design, and tests are straightforward to model and understand.

3. **Facilitation of iteration**

   - Changes are automatically transferred to the detailed diagrams of the model, reducing manual updates as much as possible.

   Note: This is based on solid planning of the model. Individual entities of the model are only described in one place.

4. **Promotion of reusability**
   - The models and components can be reused across projects, which saves time and effort.
5. **Reduction of complexity**
   - The visualization of systems through models simplifies the understanding and control of complex interactions.
6. **Improved collaboration**
   - Models act as a lingua franca, bridging gaps between disciplines and organizational units.
7. **Increased efficiency**
   - Centralized and reusable models save time and effort.
8. **Lower risk**
   - Early detection of inconsistencies and errors minimizes costly corrections in later phases.
9. **Better quality**
   - Comprehensible, validated models ensure that the systems meet the stakeholders' requirements.
10. **Future security**

- Models remain valuable documentation resources for system upgrades and maintenance.

Example: Application of MBSE

Imagine a company developing an autonomous vehicle system. Use of MBSE:

1. **System Concept Definition**:

- High-level models outline system goals such as autonomous navigation, passenger safety, and regulation compliance.
- The stakeholders model these goals visually using SysML diagrams.

2. **Requirements Modeling**:

- Detailed requirements models capture functional requirements (e.g., obstacle detection) and non-functional restrictions (e.g., system latency).
- Traceability links each requirement with system elements and test cases.

3. **Architecture definition**:

- Functional architecture models divide the system into modules (e.g., sensor integration, AI processing).
- Logical architecture models define the interactions between these modules.

4. **Detailed design:**
   - Developers create detailed designs for software and hardware components, ensuring compliance with system requirements.
   - Simulation tools validate the design at an early stage of development.

Dipl.- Ing. Stefan Peil | Zur Eiche 14 | 35096 Weimar (Lahn) | Germany

5. **Testing and validation:**

   - Test models describe how individual components and the entire system are validated.
   - Automated tools simulate tests and validate the results based on the requirements.

The team topologies: Overview

Team Topologies, a modern organizational concept, focus on cross-functional and dynamic team design with clearly defined responsibilities and communication channels instead of project management of individual human resources. It is a contemporary approach to designing, structuring, and managing teams within an organization to maximize collaboration, efficiency, and workflow.

The concept was introduced in the book Team Topologies by Matthew Skelton and Manuel Pais. It focuses on designing teams to align with technical, organizational, and value stream goals, with an emphasis on dynamics, team interactions, and responsibility for specific development areas. One goal of the Team Topology approach is to reduce teams' cognitive load through better team organization.

Instead of traditional hierarchical, departmental, or isolated team structures, the team topology approach promotes flexible, purpose-driven teams with clear boundaries, unambiguous responsibilities, and defined communication. It is crucial in complex systems such as highly networked software development or cyber-physical systems, where collaboration between different development areas is essential.

Basic principles of the team topology

1. **Team types**

   In the team topology approach, there are four basic team types, each with specific roles and responsibilities:

   - **Stream-Aligned Teams:** These teams are geared towards end-to-end development. They deliver end-to-end added value and are closest to the needs of the system's future users.
   - **Platform Teams:** Focus on reusable services, tools, or components that other teams can use to reduce cognitive load.
   - **Enabler teams:** Provide expertise or support in specific areas, such as new technology or methodology, to help other teams overcome obstacles.
   - **Core Teams:** They manage system areas that require specialized knowledge or expertise, such as complex algorithms or hardware integration.

2. **Team interactions**

   - **Collaboration:** Teams work closely on a common goal over a defined period.
   - **X-as-a-Service:** A team provides a service or product that others can use without close collaboration.
   - **Facilitating:** One team helps another team to learn or adopt a new approach or tool.

3. **Cognitive load management**

   - This approach focuses on reducing the cognitive load on teams to ensure that each team is focused on a straightforward task and not overloaded by unrelated issues.

4. **Boundaries and interfaces**

   - Clear boundaries between the teams help to define responsibilities and accountabilities, while interfaces (such as APIs or workflows) regulate the interaction between the teams.

Identified advantages of team topologies in practice:

The team topology approach is particularly valuable in modern, fast-moving, and complex systems:

1. **Breaking open silos**
   - Collaboration is encouraged beyond traditional team boundaries.
2. **Increase efficiency**
   - Teams are structured to create value and are not organized around rigid departmental roles.
3. **Strengthen adaptations and flexibility**
   - Supports the dynamic reconfiguration of teams when requirements change within development.
4. **Optimize processes**
   - Teams are aligned with the business priorities of the value stream to reduce handovers and delays.
5. **Focus on the effectiveness and motivation of the teams**
   - Ensures that teams can work effectively without being overloaded.
6. **Clear responsibilities**
   - Each team knows its tasks, goals, and responsibilities.
   - Teams interact in a predefined, efficient way, minimizing idle time and friction losses.
   - The tasks and APIs of the teams can be described in the MBSE model
7. **Fewer bottlenecks**
   - Teams only interact when necessary, avoiding unnecessary dependencies.

- **Improved provision**
  - Companies achieve faster time-to-market by focusing on the flow and delivery of business value.

Example: Application of team topologies

Imagine an organization developing a new smart home device. Using the team topology approach:

- A **stream-aligned team** focuses on providing the device's functionality (e.g., user interfaces and device connectivity).
- A **platform team** provides reusable services such as a cloud storage system or a test framework.
- An **enabling team** supports the stream-aligned team in introducing a machine-learning system for innovative functions.
- A **core team** designs a customized sensor module requiring special hardware design knowledge.

These teams interact with clear objectives and defined modes of cooperation to ensure a smooth process without unnecessary delays or misunderstandings.

Dipl.- Ing. Stefan Peil | Zur Eiche 14 | 35096 Weimar (Lahn) | Germany

Research question

How can team topologies optimize the team structure for V-model-based development of safety-critical cyber-physical systems to align with the principles of Model-Based Systems Engineering (MBSE) and reduce inefficiencies caused by traditional silo approaches and document-based development?

Why team topologies are the right approach for combining with the V-model

The main features of team topologies for this particular approach are:

1. **Improved collaboration:**
   - Stream-aligned teams focus on delivering continuous development results according to the implementation plan.
   - Teams that enable support (enabler teams) fulfill special requirements, such as MBSE expertise.
2. **Improved feedback cycles:**
   - Continuous collaboration between the development and test functions enables problems to be solved more quickly.
3. **Focus on MBSE:**
   - MBSE emphasizes a unified system model that aligns with the cross-functional ethos of team topologies.
4. **Scalability and adaptability:**
   - Modular model structures make scaling and adapting roles easier when project requirements change.
5. **System responsibility:**
   - Cross-functional stream-aligned teams maintain a system-wide perspective, ensuring the holistic integration of cyber-physical components.

Proposed solution: Integration of team topologies with the V-model and MBSE

Team tasks for development steps

To overcome the challenges of the traditional V-model and use MBSE at the same time, teams can be structured as follows:

1. **Definition of the system concept**

Team type: Stream-Aligned Team

- Tasks: Work with stakeholders, particularly senior management, to define the development vision, constraints, and specific MBSE framework.

2. **Definition of the functional architecture**

Team type: Stream-Aligned Team

- Tasks: Work with stakeholders, especially potential customers/users, to define the system's core functions.

3. **Determining the system requirements**

Team type: Stream-Aligned Team

- Tasks: Work closely with stakeholders to elicit, validate and model system requirements.

4. **Definition of the logical architecture**

Team type: Stream-Aligned Team and Core Team

- Tasks: Provides reusable logical model-based components and supports functional teams with architectural consistency.

5. **Structure of the cyber-physical architecture with concrete solutions**

Team type: Core team

- Tasks: Design the concrete system architecture, including the cyber and physical domains.

6. **Definition of software and hardware specifications**

Team type: Core team

- Tasks: Develop domain-specific specifications that ensure compatibility with system-level models.

7. **Translation of specifications into specific component design**

Team type: Core team with support from the enabler team and platform team

- Tasks: Implement detailed component designs using MBSE tools and developer suites to maintain traceability and alignment.

8. **Carrying out component tests**

Team type: Core team

- Tasks: Design test frameworks and ensure compliance with safety and performance standards.

9. **Integration and integration tests**

Team type: Core team

- Tasks: Performs integration, testing, and aligning subsystems with overarching system models.

10. **System assembly**

Team type: Core team, platform, and enabler team

- Tasks: Lead the end-to-end system assembly and ensure the methods and tools for final validation. The platform team provides suitable test suites

11. **Validation of the system concept**

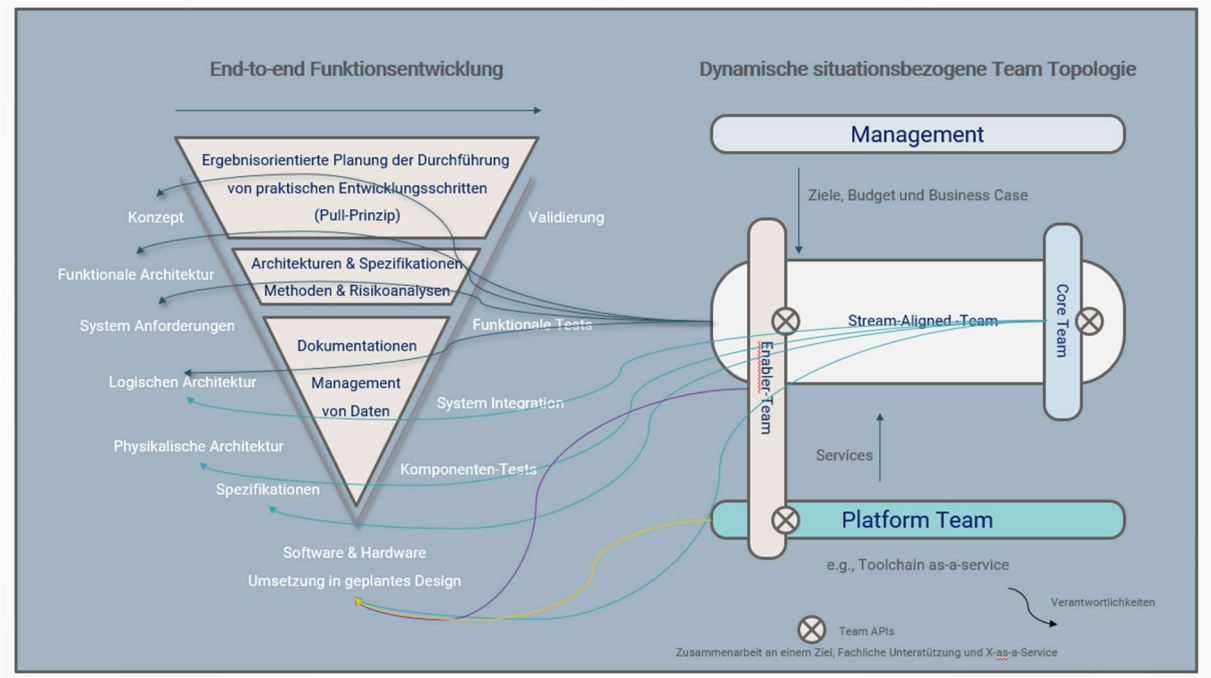
Team type: Stream-aligned team with support from the core team

- Tasks: Work with stakeholders and regulators to validate and approve the system based on its initial concept.

Supporting structures preferably managed by the platform team

1. **Cross-divisional roles**
   - Creating team communication models to facilitate communication between Stream-Aligned, Enabler and Core Teams (Team APIs).
2. **Tools and automation**
   - Use of MBSE tools such as SysML to ensure traceability.
   - Automation of development and validation processes.
3. **Training and further education**
   - Imparting cross-functional knowledge to team members to promote collaboration and modeling

The following graphic serves as a blueprint for the design and composition of the team topology:

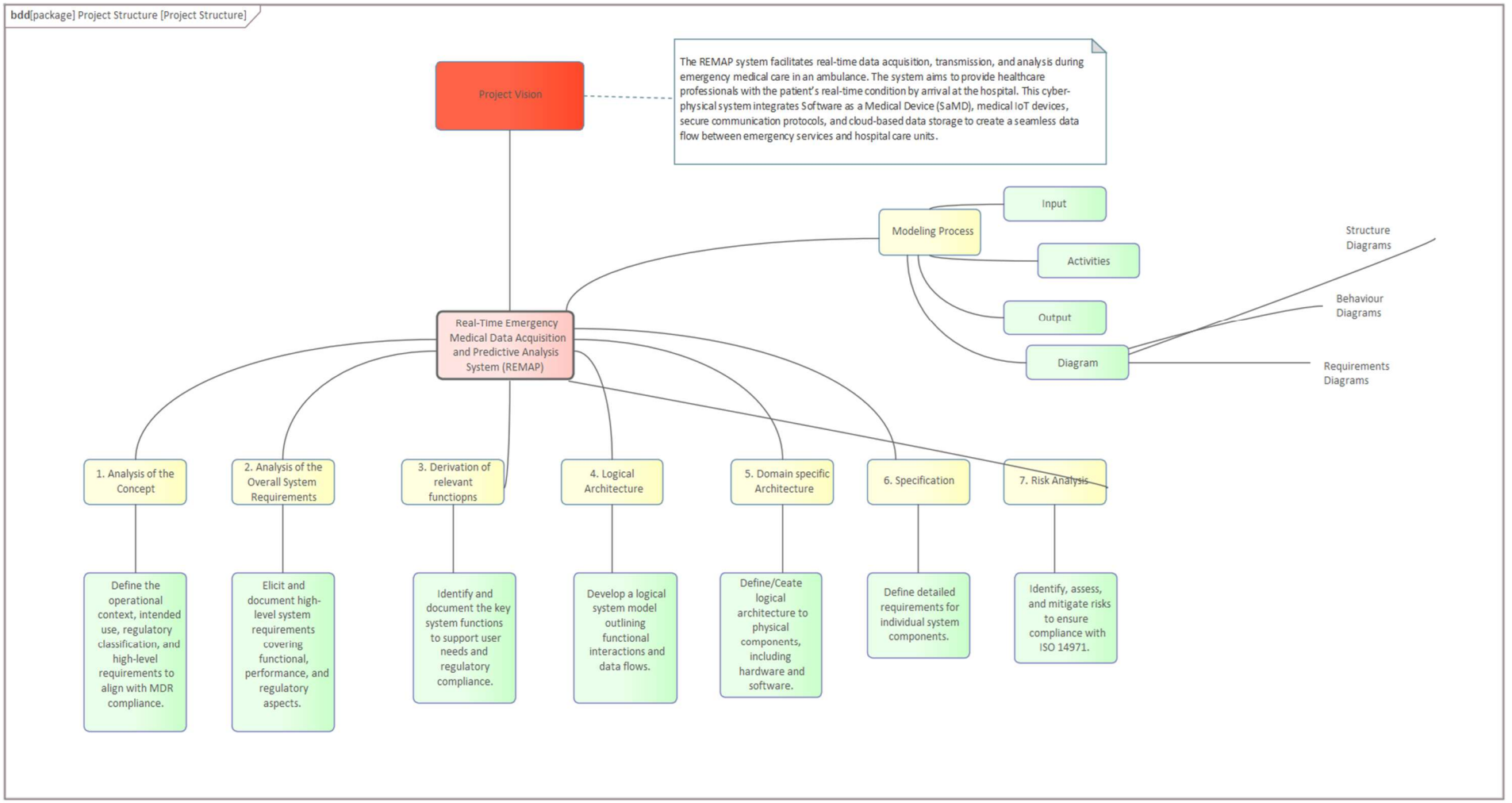


Example application and visualization with the MBSE tool SPARX Enterprise Architect

The "Real-Time Emergency Medical Data Acquisition and Predictive Analysis System (REMAP)"

The REMAP system enables the collection, transmission, and analysis of patient data in real-time during emergency medical care in an ambulance. The system aims to communicate the patient's condition to medical staff in real-time upon arrival at the hospital. This cyber-physical system integrates Software as a Medical Device (SaMD), medical IoT devices, secure communication protocols, and cloud-based data storage to create a seamless data flow between ambulance services and hospital wards.

Project Structure



The figure shows the project structure for the Real-Time Emergency Medical Data Acquisition and Predictive Analysis System (REMAP). It describes the vision of the system, which records, transmits, and analyzes medical data in emergency vehicles in real time. The aim is to provide medical professionals with information about the patient's condition before they arrive at the hospital. The system combines software as a medical device (SaMD), medical IoT devices, secure communication protocols, and cloud storage solutions.

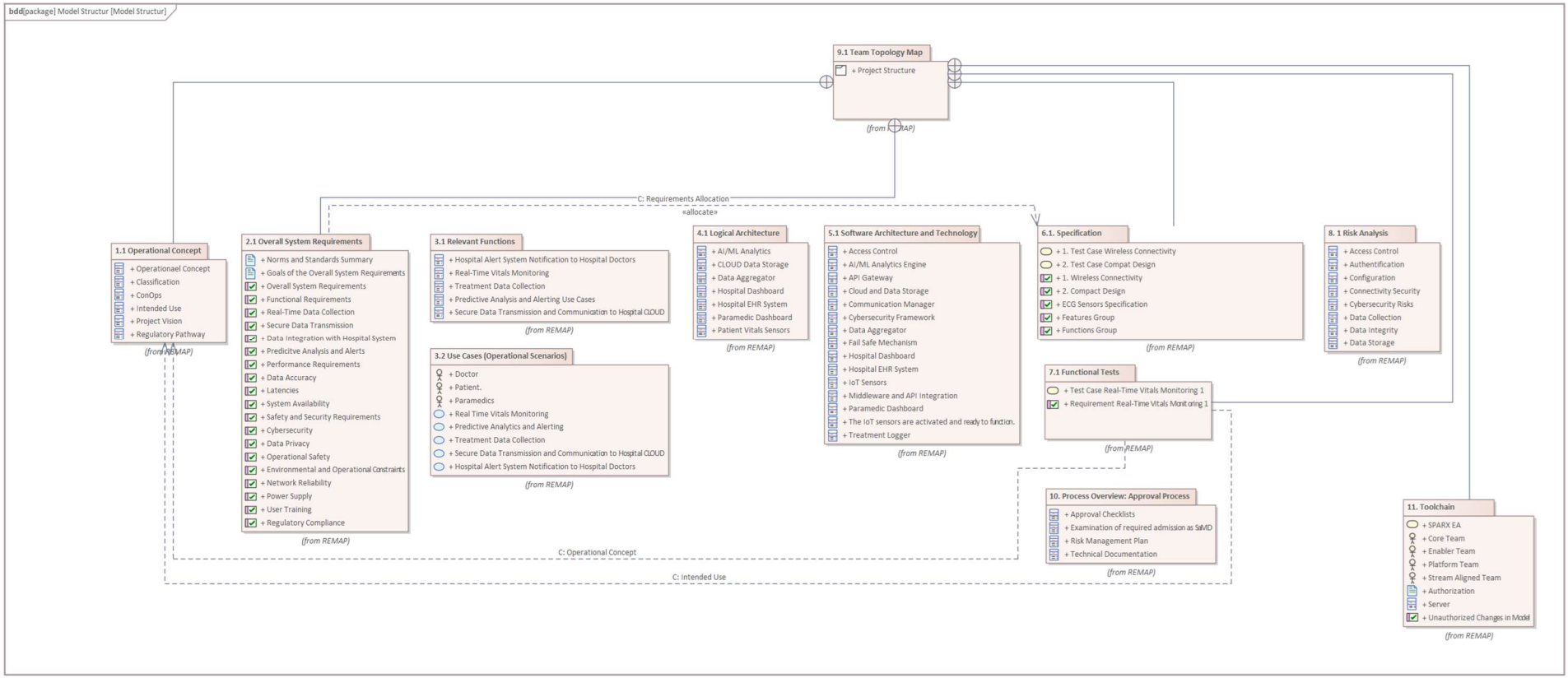
The diagram structure divides the project work into seven main areas:

1. Analysis of the concept (definition of the use case and regulatory framework).
2. Analysis of the system requirements (functional, performance-related, and regulatory).
3. Derivation of relevant functions (user and compliance requirements).
4. Logical architecture (functional interactions and data flows).
5. Domain-specific architecture (hardware and software components).
6. Specification (detailed requirements).
7. Risk management (compliance with ISO 14971).

In addition, the modeling process is divided into inputs, activities, outputs, and diagram types (structure, behavior, and requirements diagrams).

Note: The system is not shown in full for all illustrations due to intellectual property protection.

Model Structure

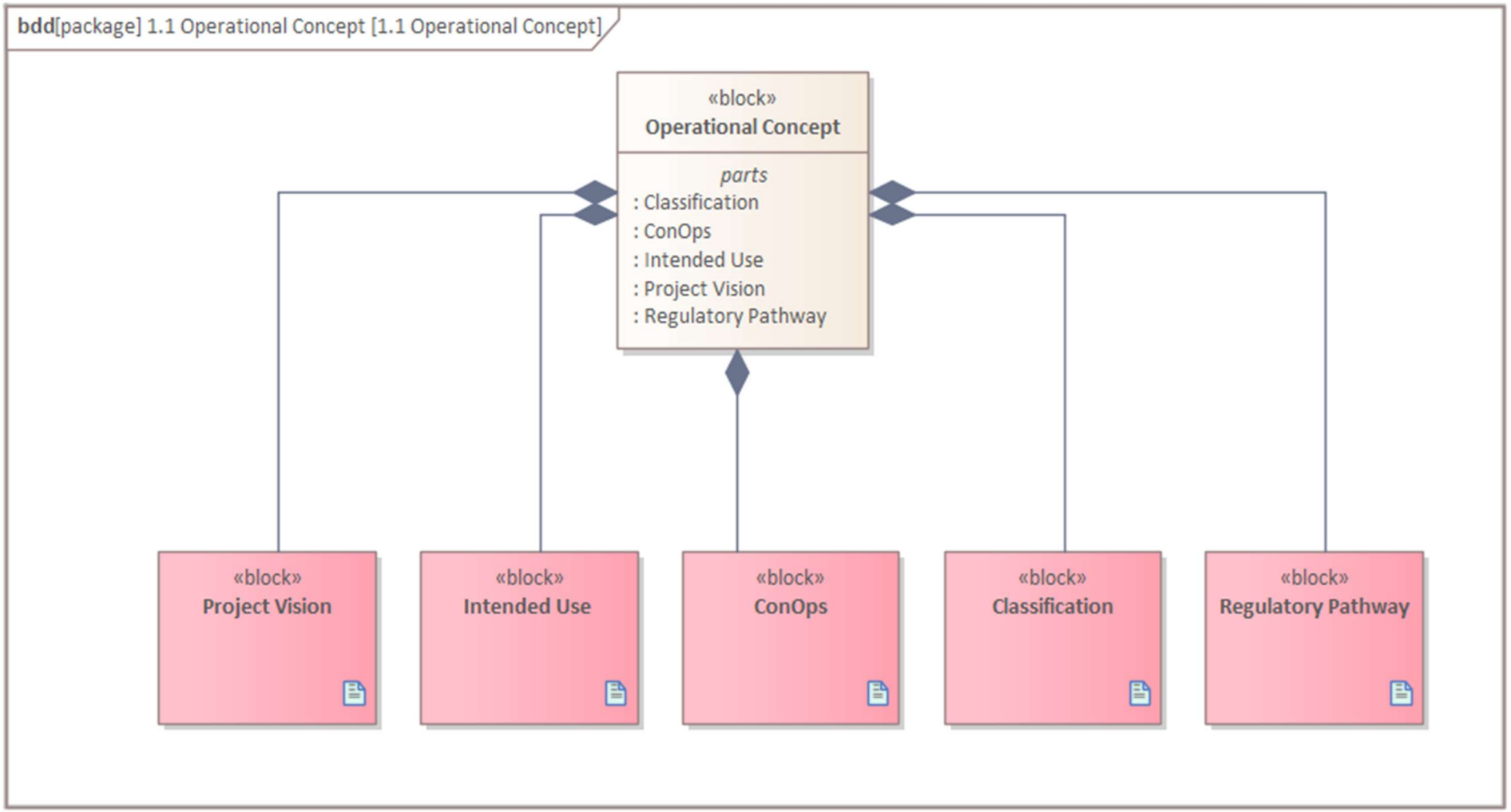


The figure shows the model structure for the **Real-Time Emergency Medical Data Acquisition and Predictive Analysis System (REMAP)** project. It illustrates the relationships and dependencies between different elements of the system. The structure comprises several main areas:

1. **Operational Concept (1.1):** Describes the concept, including classification, planned use, vision, and regulatory pathways.
2. **Overall System Requirements (2.1):** This section defines functional, security, regulatory, and operational requirements, such as real-time data collection, cyber security, and user training.
3. **Relevant Functions (3.1):** Key functions such as hospital notifications, real-time monitoring of vital data, and secure data transmission.
4. **Use cases (3.2):** Scenarios for doctors, patients, and paramedics, e.g., monitoring and analysis of vital data.
5. **Logical Architecture (4.1):** Components such as AI/ML analytics, cloud data storage, dashboards, and vital data sensors.
6. **Software Architecture and Technology (5.1):** Technological foundations like API gateways, cybersecurity frameworks, and IoT sensors.
7. **Specification (6.1):** Specifications for wireless connectivity, sensor design and functions.
8. **Functional tests (7.1): Functional requirements** and assigned
9. **Risk Analysis (8.1):** This includes risks such as access control, cyber security, and data protection.
10. **Team Topology Map (9.1):** Team structure and project assignment.
11. **Process Overview: Approval Process (10):** This section provides an overview of approval processes, including risk management and technical documentation.
12. **Toolchain (11):** Tools like SPARX EA and Teams for various tasks.

The figure provides an overview of the modeling, system integration, and regulatory requirements of the REMAP system.

Operational Concept



The figure shows the structure of the **Operational Concept (1.1)** for the REMAP project. It describes the main components required to design and understand the system. The individual blocks include:

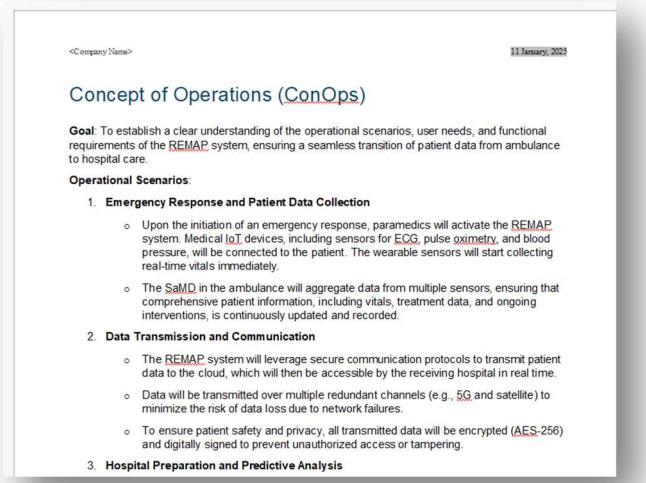
1. **Project Vision:** Defines the overall vision and goals of the project, including the intended benefits for end users and stakeholders.
2. **Intended Use:** Describes the intended use of the system, including the target environment and the expected functionality.
3. **ConOps (Concept of Operations):** Sets out how the system will be used in an operational context, including use cases and scenarios.
4. **Classification:** Assigns the system to a regulatory and functional classification to ensure compliance with legal requirements.
5. **Regulatory Pathway:** Outlines the regulatory approval pathway, including the relevant norms and standards that must be met.

This structure serves as the basis for anchoring the system's requirements, regulatory specifications, and operational objectives in a clearly defined concept.

The details of the concept are described in anchored documents:

Example of an anchored document

in SPARX EA



<Company Name> 11 January, 2025

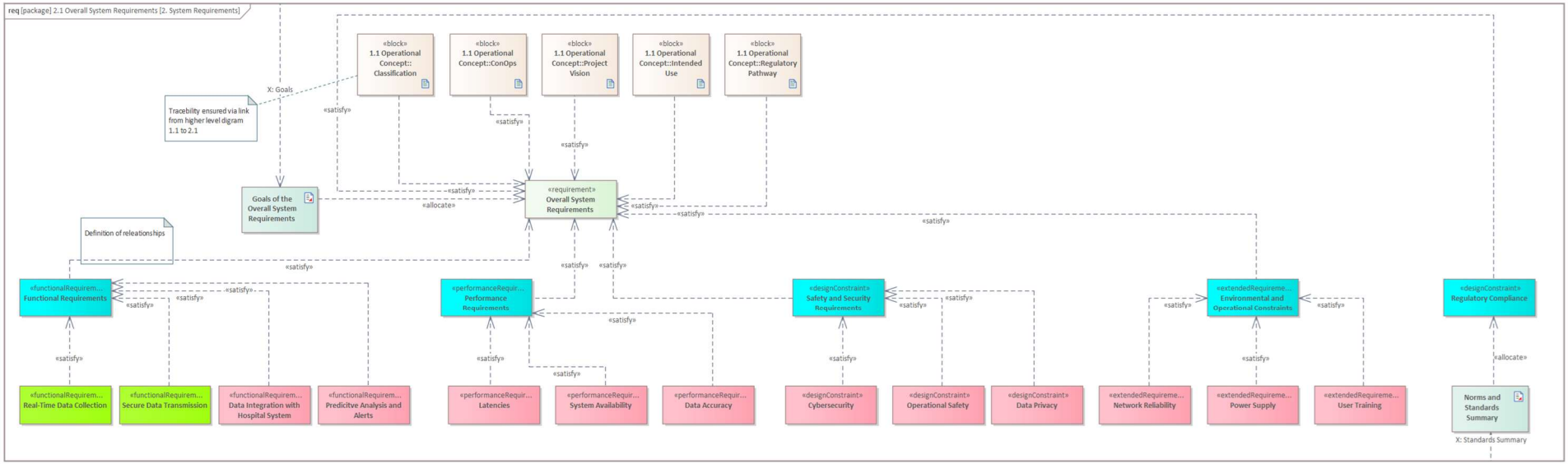
Concept of Operations (ConOps)

Goal: To establish a clear understanding of the operational scenarios, user needs, and functional requirements of the REMAP system, ensuring a seamless transition of patient data from ambulance to hospital care.

Operational Scenarios:

1. **Emergency Response and Patient Data Collection**
   - Upon the initiation of an emergency response, paramedics will activate the REMAP system. Medical IoT devices, including sensors for ECG, pulse oximetry, and blood pressure, will be connected to the patient. The wearable sensors will start collecting real-time vitals immediately.
   - The SaMD in the ambulance will aggregate data from multiple sensors, ensuring that comprehensive patient information, including vitals, treatment data, and ongoing interventions, is continuously updated and recorded.
2. **Data Transmission and Communication**
   - The REMAP system will leverage secure communication protocols to transmit patient data to the cloud, which will then be accessible by the receiving hospital in real time.
   - Data will be transmitted over multiple redundant channels (e.g., 5G and satellite) to minimize the risk of data loss due to network failures.
   - To ensure patient safety and privacy, all transmitted data will be encrypted (AES-256) and digitally signed to prevent unauthorized access or tampering.
3. **Hospital Preparation and Predictive Analysis**

System Requirements

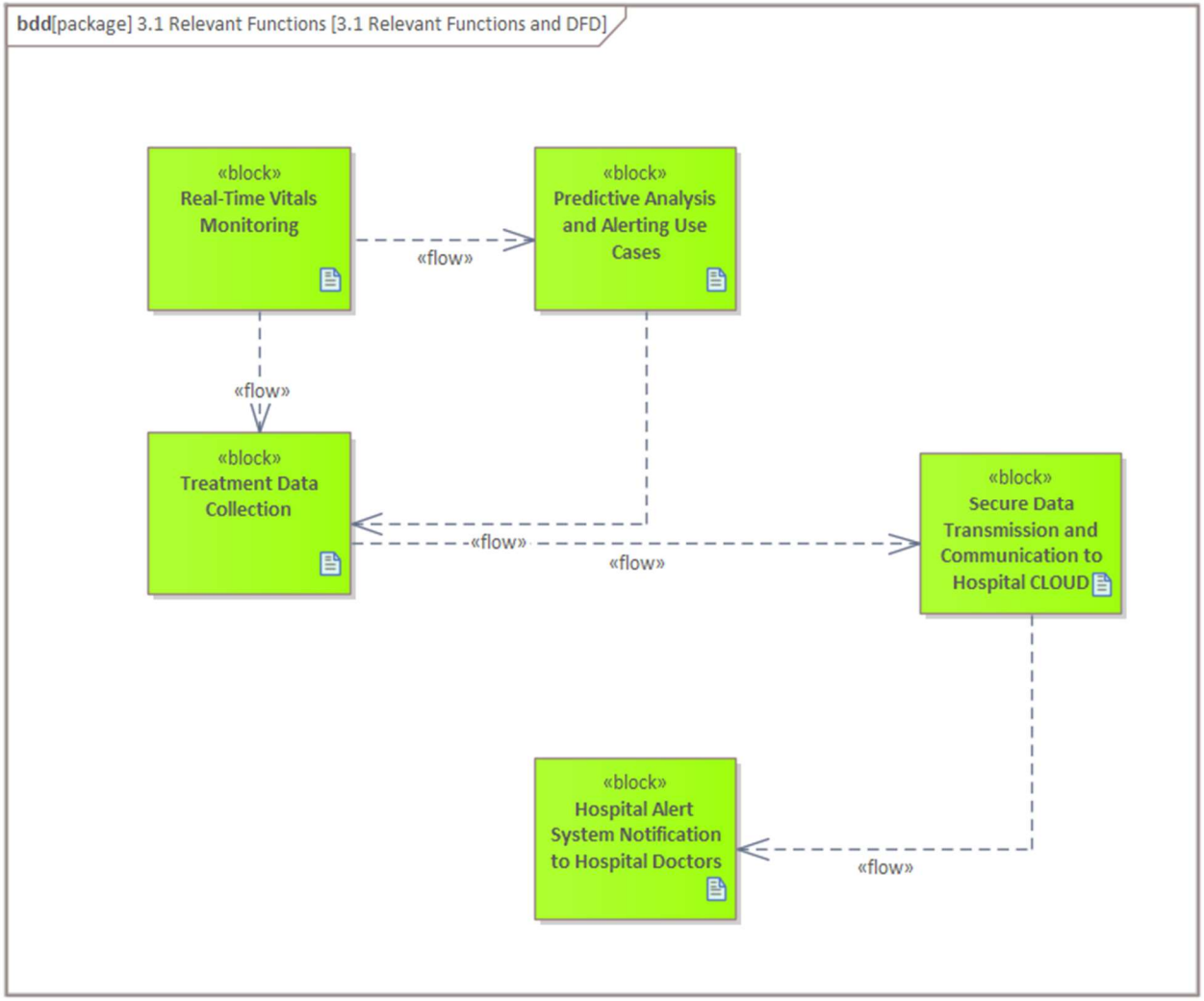


The figure shows the **system requirements (2.1 Overall System Requirements)** of the REMAP project and illustrates the traceability and links between different requirements and concepts. It includes:

1. **Objectives of the overall requirements:** This section describes the overarching goals and derives from higher-level concepts such as the Operational Concept (1.1), including Project Vision, Intended Use, ConOps, Classification, and Regulatory Paths.
2. **Functional requirements:** Includes specific system functions such as:
    - **Real-time data acquisition**
    - **Secure data transmission**
    - **Integration with hospital systems**
    - **Predictive analyses and warnings**
3. **Performance requirements:** Ensures the system meets latency, availability, and data accuracy requirements.
4. **Constraints:** These include security, cyber security, data protection, and operational security requirements.
5. **Extended requirements refer to operational and environmental constraints such as power supply, network reliability,** and user training.
6. **Regulatory compliance:** Ensures compliance with relevant norms and standards and links these to the corresponding requirements.

The diagram provides an overview of how functional, performance-related, and regulatory requirements are met and how these can be traced back to the higher system objectives. It supports traceability and facilitates the system's validation and verification.

Relevant Core Functions

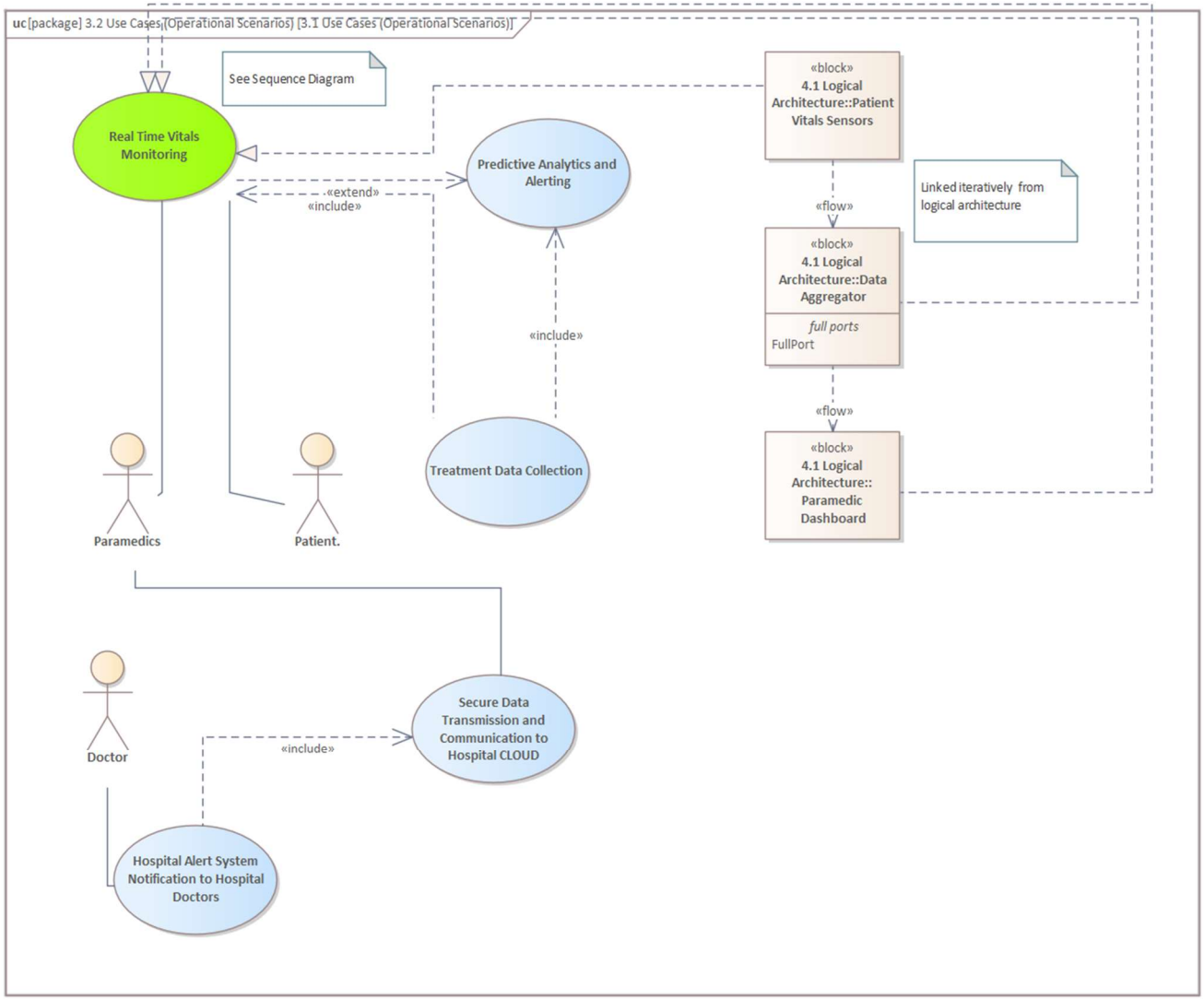


The figure shows the **relevant core functions (3.1 Relevant Functions)** of the REMAP system and their data flows. These functions represent the core processes of the system that interact with each other to enable the real-time collection, analysis, and transmission of medical data. The most important blocks are:

1. **Real-Time Vitals Monitoring:** This monitors the patient's vital signs, such as heart rate and oxygen saturation, in real time.
2. **Treatment Data Collection:** Collects and documents the data obtained during treatment.
3. **Predictive Analysis and Alerting Use Cases:** Performs predictive analyses to detect emergencies at an early stage and generate corresponding alerts.
4. **Secure Data Transmission and Communication to Hospital CLOUD:** The recorded and analyzed data are securely transmitted to the hospital cloud to ensure access for medical staff.
5. **Hospital Alert System Notification to Hospital Doctors:** The hospital alert system sends notifications with relevant patient data and analyses directly to hospital doctors to enable rapid decision-making.

The **data flows** shown illustrate the relationships and interactions between the functions and clarify the sequence and dependencies within the data processing chain. This diagram supports the visualization of the operational logic of the REMAP system.

Use Cases

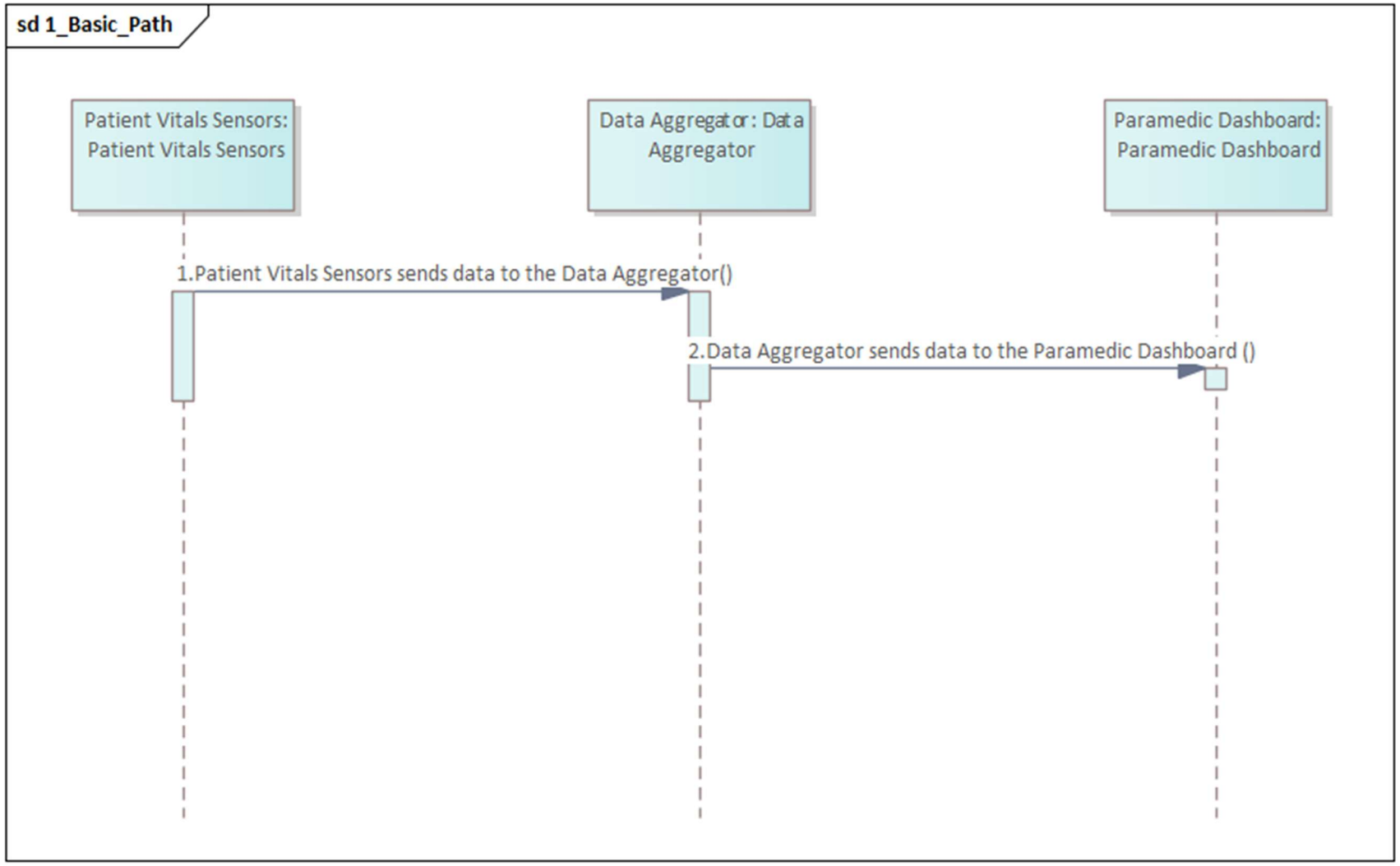

uc [package] 3.2 Use Cases (Operational Scenarios) [3.1 Use Cases (Operational Scenarios)]
See Sequence Diagram
Real Time Vitals Monitoring
Predictive Analytics and Alerting
«extend»
«include»
«include»
«block»
4.1 Logical Architecture::Patient Vitals Sensors
«flow»
Linked iteratively from logical architecture
«block»
4.1 Logical Architecture::Data Aggregator
full ports
FullPort
«flow»
«block»
4.1 Logical Architecture:: Paramedic Dashboard
Treatment Data Collection
Paramedics
Patient.
Doctor
«include»
Secure Data Transmission and Communication to Hospital CLOUD
Hospital Alert System Notification to Hospital Doctors

The figure shows the **use cases (*3.2* Operational Scenarios)** of the REMAP system, which visualize the interactions between the actors (paramedics, patients, and doctors) and the system's main functions. The scenarios shown describe how the system is used in an emergency operation:

1. **Real-Time Vitals Monitoring:**
    - Paramedics monitor the patient's vital signs in real-time using sensors.
    - This includes **Patient Vitals Sensors**, which collect the data, and the **Data Aggregator**, which processes and forwards the data.
2. **Predictive Analytics and Alerting:**
    - Expansion of vital data monitoring through predictive analyses to detect critical conditions.
    - Notifications are generated in emergencies and forwarded to medical staff.
3. **Treatment Data Collection:**
    - Captures the data collected during treatment and integrates it into the analysis.
4. **Secure Data Transmission and Communication to Hospital CLOUD:**
    - Transfers vital and treatment data securely to the hospital cloud to make it available to doctors.
5. **Hospital Alert System Notification to Hospital Doctors:**
    - Hospital doctors receive real-time notifications and can prepare for the patient's arrival.

The illustration also links the **logical architecture components**, such as the **Paramedic Dashboard**, which provides paramedics with a user-friendly display of the collected data. This ensures that all use cases are integrated into the system logic and data flows.

Sequence Diagram, generated from the Use Case (example: Real-Time Vitals Monitoring)



The sequence diagram shows the **basic data flow (Basic Path)** in the REMAP system, which illustrates the communication between the three main components:

1. **Patient Vitals Sensors:**
    - The sensors monitor the patient's vital signs and send this data to the **data aggregator**.
2. **Data Aggregator:**
    - The Data Aggregator processes the incoming vital data and prepares it for display on the **Paramedic Dashboard**.
3. **Paramedic Dashboard:**
    - The processed data is displayed on the dashboard to give the paramedics an overview of the patient's vital signs in real time.

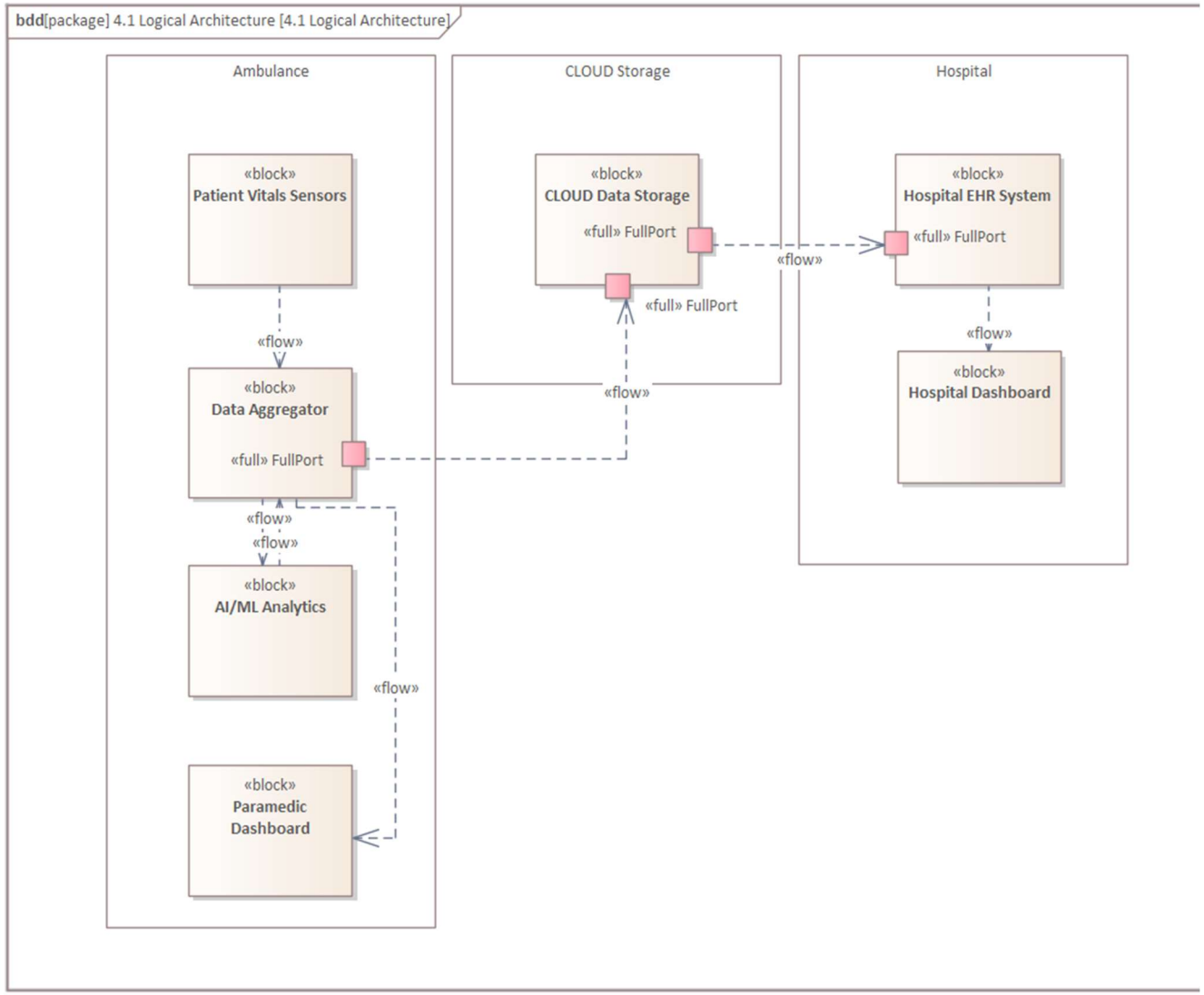
The process is divided into two main steps:

- **Step 1:** The patient sensors send the vital data to the aggregator.
- **Step 2:** The Data Aggregator transmits the processed data to the Paramedic Dashboard.

This diagram illustrates the interaction between hardware (sensors), software (aggregator), and the user interface (dashboard) to ensure efficient data transmission and display in real-time.

The diagram can be generated directly from a use case diagram with SPARX EA.

Logical Architecture



The figure shows the **logical architecture (4.1 Logical Architecture)** of the REMAP system and describes the data flows between the main components, which are divided into three main areas:

1. Ambulance (vehicle components):

- **Patient Vitals Sensors:** record the patient's vital data in real-time.
- **Data Aggregator:** Aggregates and prepares the sensor data for further processing.
- **AI/ML Analytics:** Performs predictive analytics on the aggregated data to detect emergency patterns and generate alerts.
- **Paramedic Dashboard:** Visualizes vital data and analysis results to provide rescuers with information to support decision-making.

2. CLOUD Storage:

- **CLOUD Data Storage:** This system stores the data transmitted by the ambulance and securely makes it available for retrieval by the hospital.

3. Hospital (hospital components):

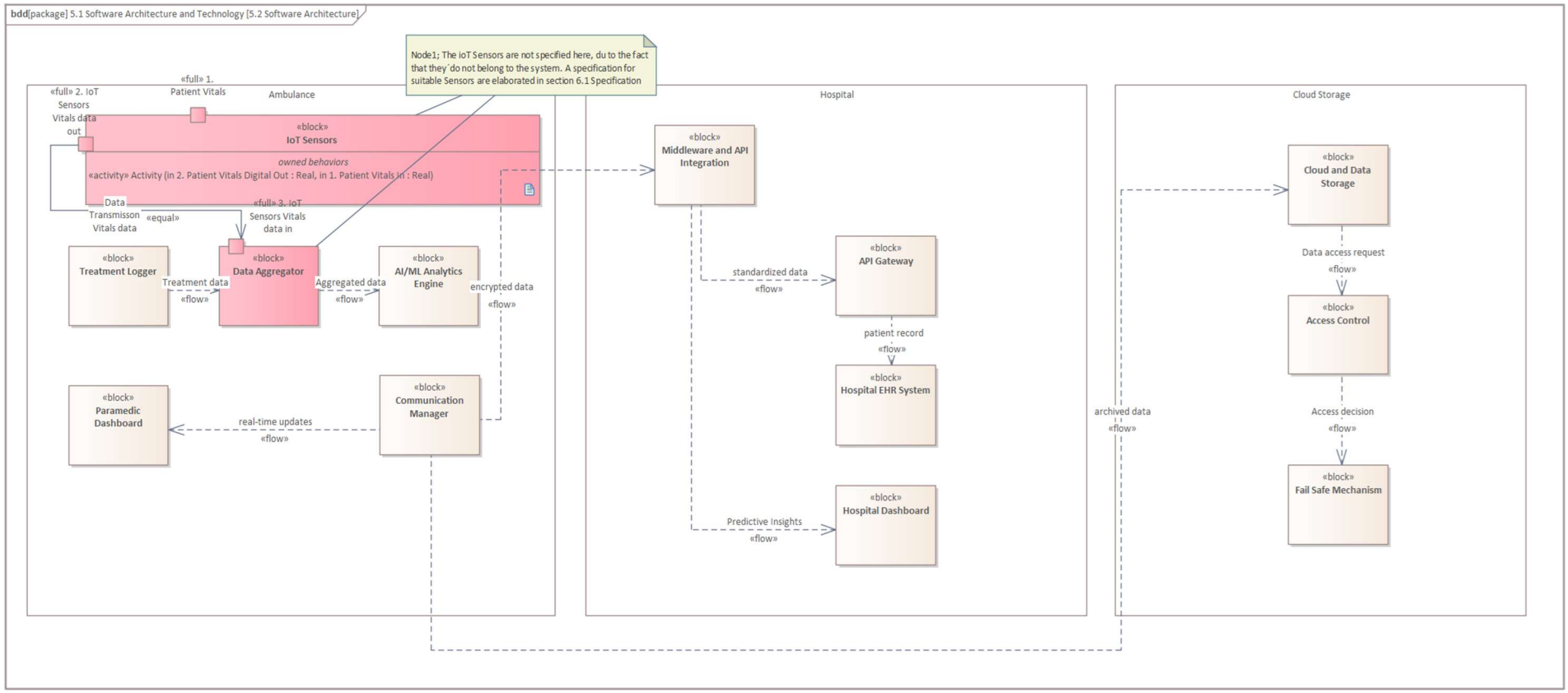
- **Hospital EHR system:** This system receives data from the CLOUD and integrates it into the electronic health record (EHR) system.
- **Hospital Dashboard:** Provides hospital physicians with an overview of patient data to prepare them for arrival.

Data flows:

- Data flows from the **Patient** Vitals Sensors via the Data Aggregator and AI/ML Analytics to the CLOUD Data Storage.
- From there, the data is forwarded to the hospital EHR system and displayed on the hospital dashboard.
- At the same time, emergency services receive real-time information on the **Paramedic Dashboard**.

This architecture demonstrates seamless and secure transfer of patient data from the ambulance to the hospital, supported by predictive analytics and real-time monitoring.

Software Architecture



The figure shows the **software architecture (5.1 Software Architecture and Technology)**, which describes the interaction of the software components for the collection, processing, storage, and transmission of patient data. The architecture is divided into three main areas:

1. Ambulance (vehicle software):

- **IoT sensors:** Record the patient's vital data and forward it to the data aggregator.
- **Data Aggregator:** Consolidates the data from the sensors and transfers it to the AI/ML Analytics Engine for further analysis or storage.
- **AI/ML Analytics Engine:** This engine performs predictive analyses of aggregated data and generates insights that are helpful for emergency decisions.
- **Treatment Logger**: Saves treatment data recorded on site.
- **Communication Manager:** Enables real-time communication and forwarding of data to cloud and hospital components.
- **Paramedic Dashboard:** Visualizes real-time data and analyses for paramedics.

2. Hospital (hospital software):

- **Middleware and API Integration:** Processes the incoming data and ensures standardization.
- **API gateway:** Mediates the data between cloud systems and the hospital.
- **Hospital EHR System:** Stores the data in the electronic patient record.
- **Hospital Dashboard:** Displays the processed data and analyses for doctors to enable quick decisions.

3. Cloud storage (cloud software):

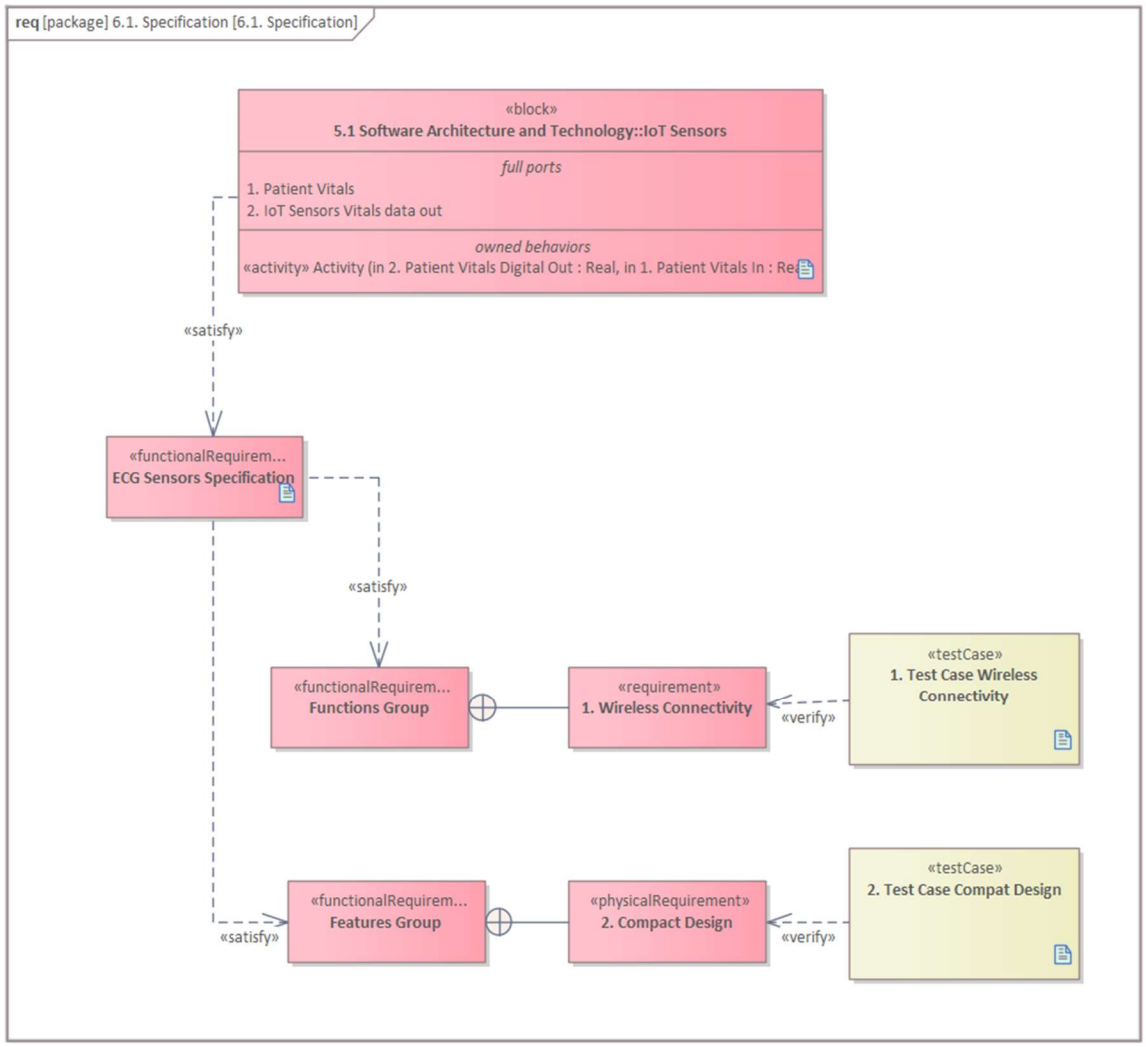
- **Cloud and Data** Storage: Central storage location for aggregated and archived data transferred from the outpatient clinic.
- **Access Control:** Ensures that data access is restricted to authorized users.
- **Fail Safe Mechanism:** Ensures data integrity and access in emergencies.

Data flows:

- Ambulance→ Cloud: Vital and aggregated data are transferred to the cloud.
- Cloud→ Hospital: Standardized data and predictive analyses are transferred to the hospital.
- Hospital→ Dashboard: Doctors access processed patient data via the dashboard.

This architecture integrates the system components from data acquisition to decision support.

Excerpt from the specification



The figure shows part of the **specification (6.1 Specification)** for the IoT sensors of the REMAP system, particularly the requirements for functionality and design and the associated test cases. The specification describes the relationships between requirements, functional groups, and verification. The most important components are

1st IoT Sensors:

- Described as part of Software Architecture and Technology (5.1).
- Responsible for the recording and output patient data (vital parameters).

2. Requirements:

- **ECG Sensors** Specification: The sensors must meet specific requirements to record vital signs such as heart rate and electrical activity.
- Functions Group: Defines the required functions, such as wireless connectivity.
- Features Group: Describes the physical features, e.g., compact design.
- Wireless connectivity: A functional requirement ensuring the sensors communicate wirelessly with other systems.
- Compact design: A physical requirement that ensures portability and ease of use.

3. Verification:

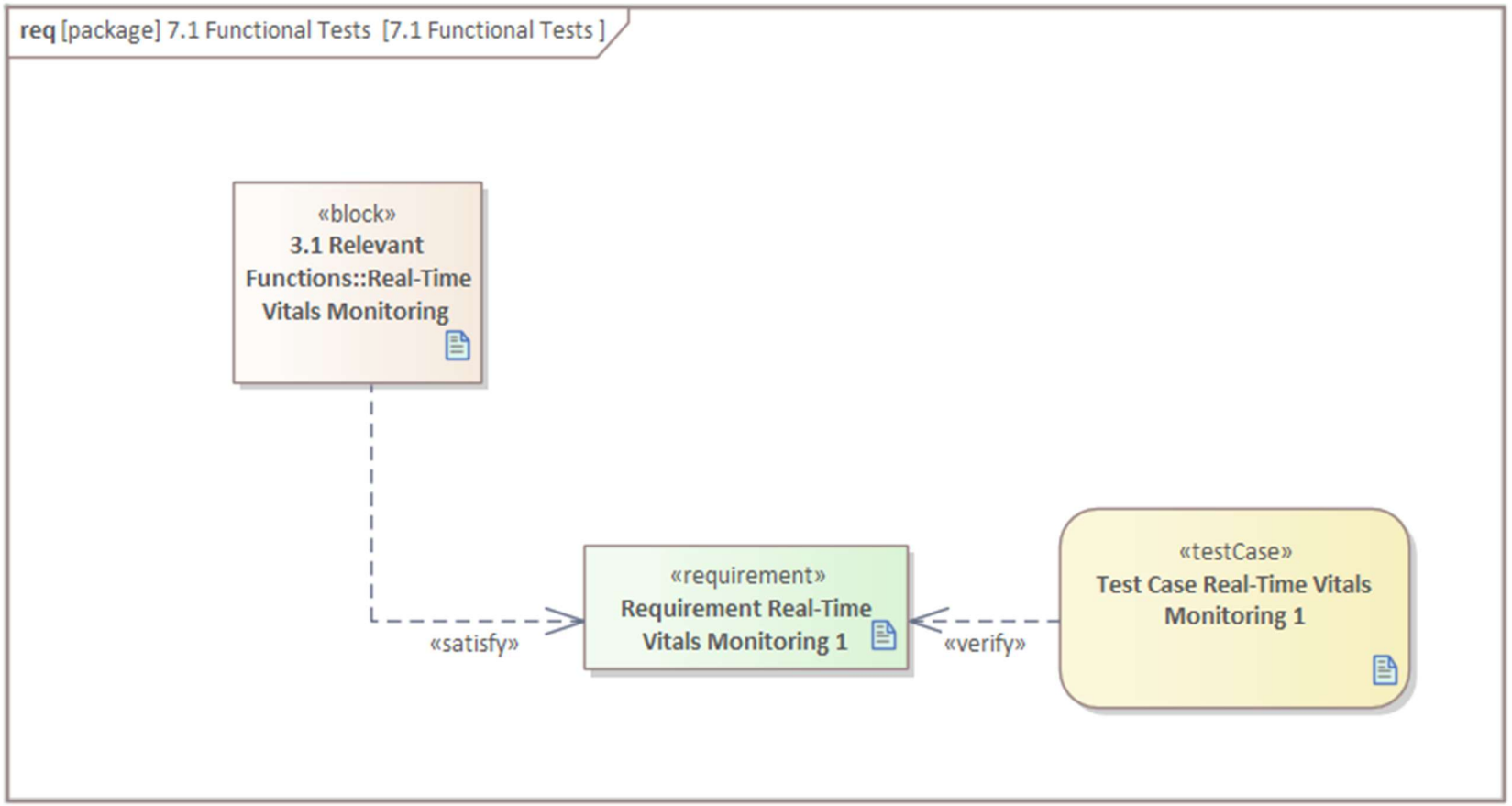
- **Test Case Wireless Connectivity:** A test case for checking the wireless communication capability of the sensors.
- Test Case Compact Design: Test case to validate compliance with the design requirements.

Data flow:

- The IoT sensors comply with the **ECG** Sensors Specification, supporting the Functions Group and Features Group.
- These requirements are verified by specific test cases, which ensure that the sensors fulfill the functional and physical requirements.

The figure shows an overview of the hierarchy and dependencies between specifications, functional requirements, and verification steps that ensure optimal sensor integration into the REMAP system.

Functional Tests



The figure shows the **functional tests (7.1 Functional Tests)** for monitoring vital data in real-time in the REMAP system. The focus is on the link between the relevant functions, the requirements, and the test cases that support implementation and verification.

Elements:

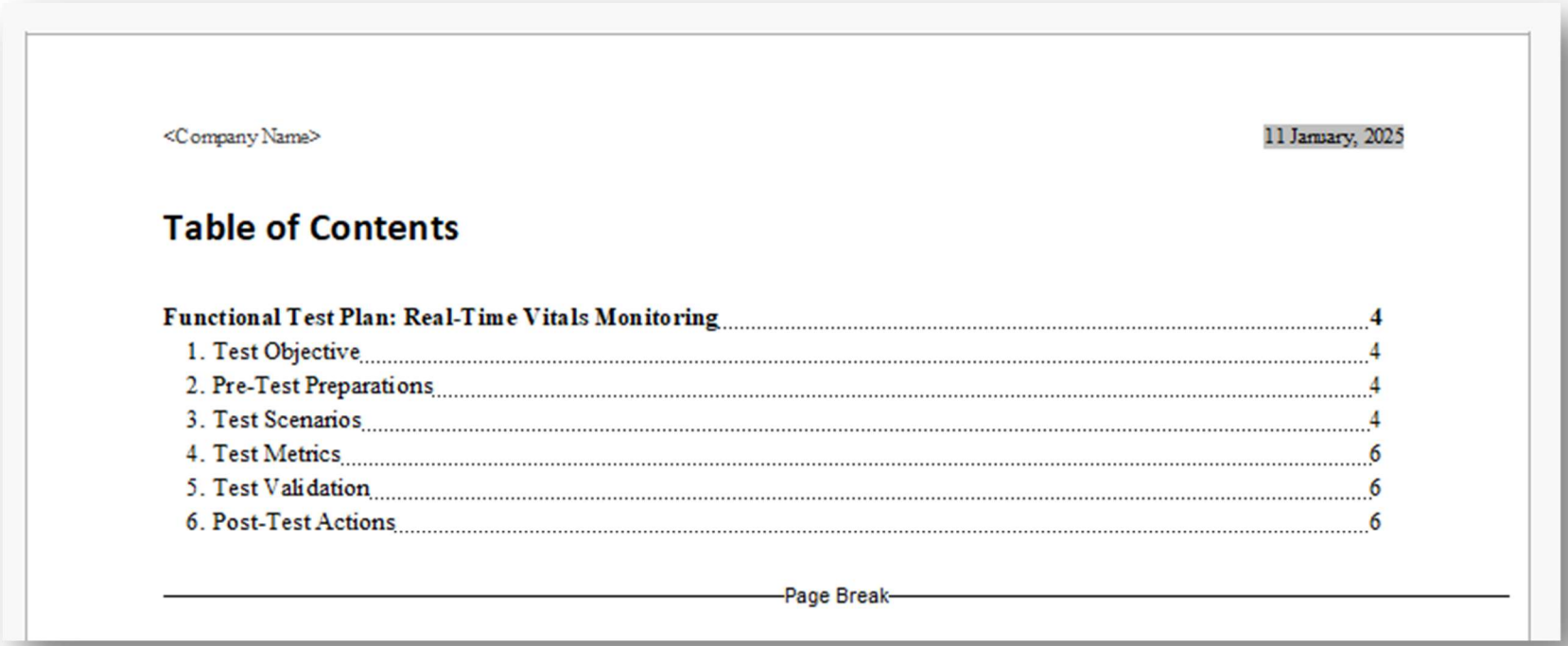
1. **Relevant Functions:**
    - **Real-Time Vitals Monitoring (3.1):** This is the main function that ensures continuous real-time monitoring of vital signs such as heart rate, blood pressure, or oxygen saturation.
2. **Requirement:**
    - **Requirement Real-Time Vitals Monitoring 1:** This functional requirement specifically defines the system's ability to monitor and provide vital data in real-time. It ensures that the system meets the operational requirements.
3. **Test-Case:**
    - **Test Case Real-Time Vitals Monitoring 1:** This is a specific test case developed to verify that the requirement for real-time monitoring of vital signs is met. It ensures verification and is essential for validating the system functionality.

Contexts:

- The **requirement** for real-time monitoring is supported by the **relevant function** and verified by the **test case**.
- The test case ensures compliance between the system performance and the defined requirements.

This structured connection shows how the real-time monitoring function can be correctly implemented and verified, which is crucial for the integrity and reliability of the REMAP system.

Extract from the anchored test case document in SPARX EA:

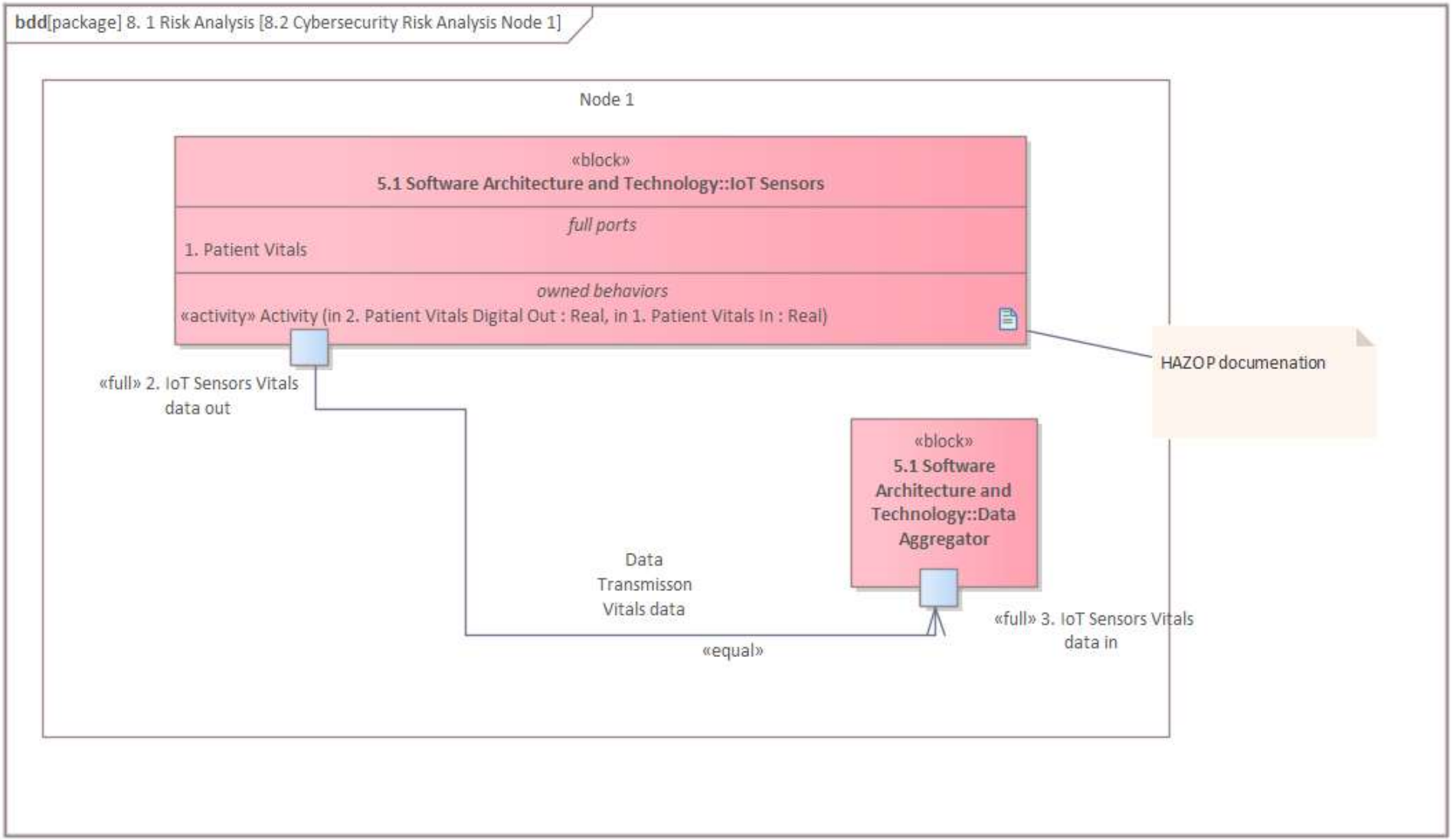


<Company Name> 11 January, 2025

Table of Contents

Page Break

Risk Analysis



The figure shows a **risk assessment (8.1 Risk Analysis)** for the data flow between IoT sensors and the data aggregator in the REMAP system. The focus is on identifying and analyzing potential risks, particularly about cyber security and data integrity.

Main components:

1. **IoT Sensors (5.1 Software Architecture and Technology):**
    - Record the patient's vital signs (e.g., heart rate, oxygen saturation) and transmit them to the data aggregator.
    - Interfaces: The data is sent to the aggregator via a defined port (full port).
2. **Data Aggregator (5.1 Software Architecture and Technology):**
    - Consolidates the received vital data and prepares it for further processing (e.g., predictive analyses).
    - Interfaces: Receiving data via the corresponding port (IoT Sensors Vitals data in).
3. **HAZOP documentation:**
    - The **Hazard and Operability Study (HAZOP)** is a systematic method for identifying potential hazards and deviations in the data flow (e.g. incomplete data transmission, incorrect data).

Data flow:

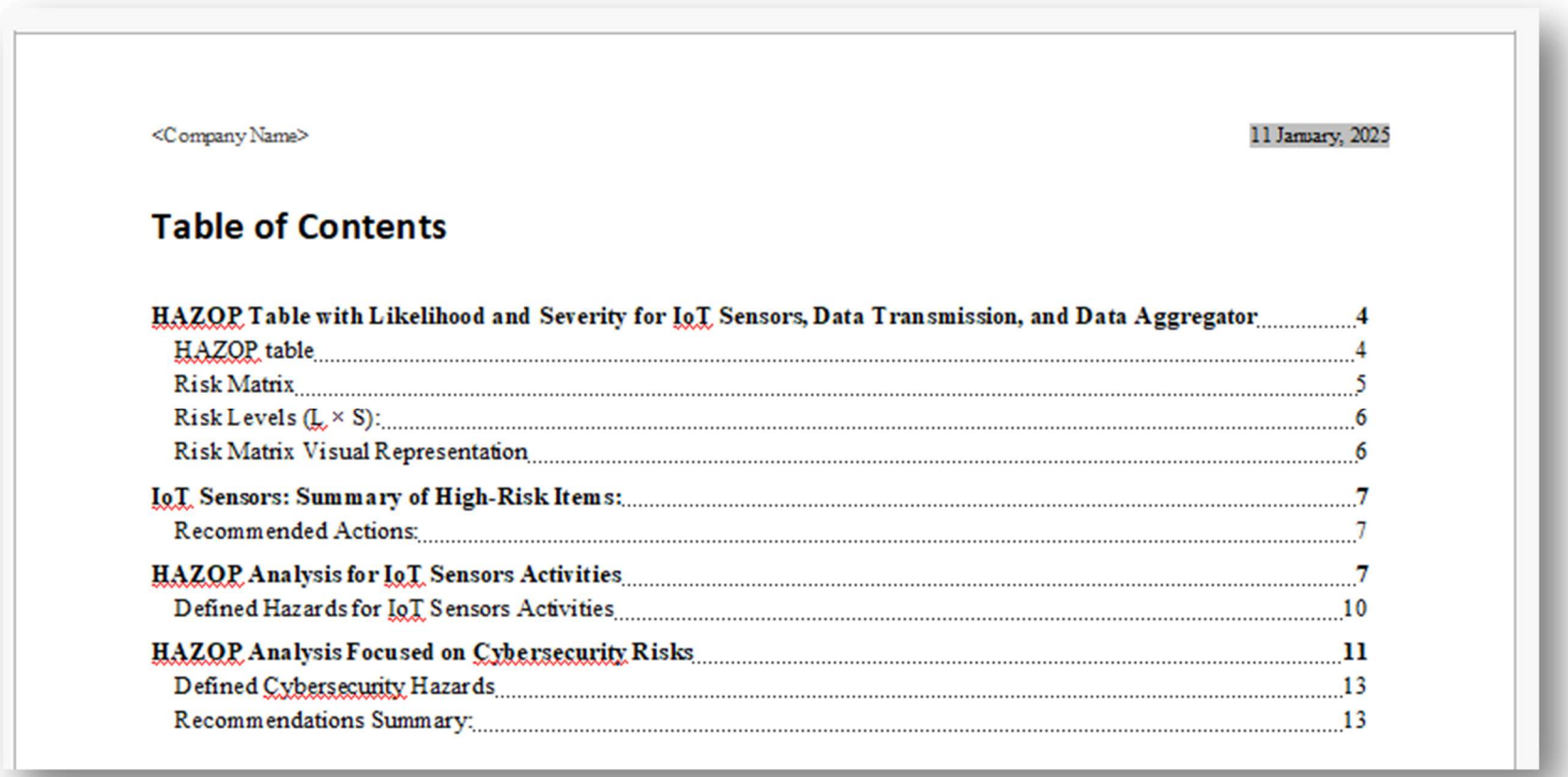
- **Data transmission from the IoT sensors to the data aggregator:** The patient data is transmitted in real-time. The data flow is described with the keyword "equal," which ensures equivalent processing of the transmitted data in the aggregator.

Purpose of the analysis:

- Ensuring **data integrity** during transmission.
- Identification of **cyber security risks**, such as unauthorized access, data loss, or manipulation during transmission.
- Documentation of potential hazards in the HAZOP documentation, which is used for safety assessment and error prevention.

This analysis is the basis for security measures and ensures that the data flow between sensors and aggregators is reliable, secure, and compliant with regulations.

Example of the HAZOP analysis

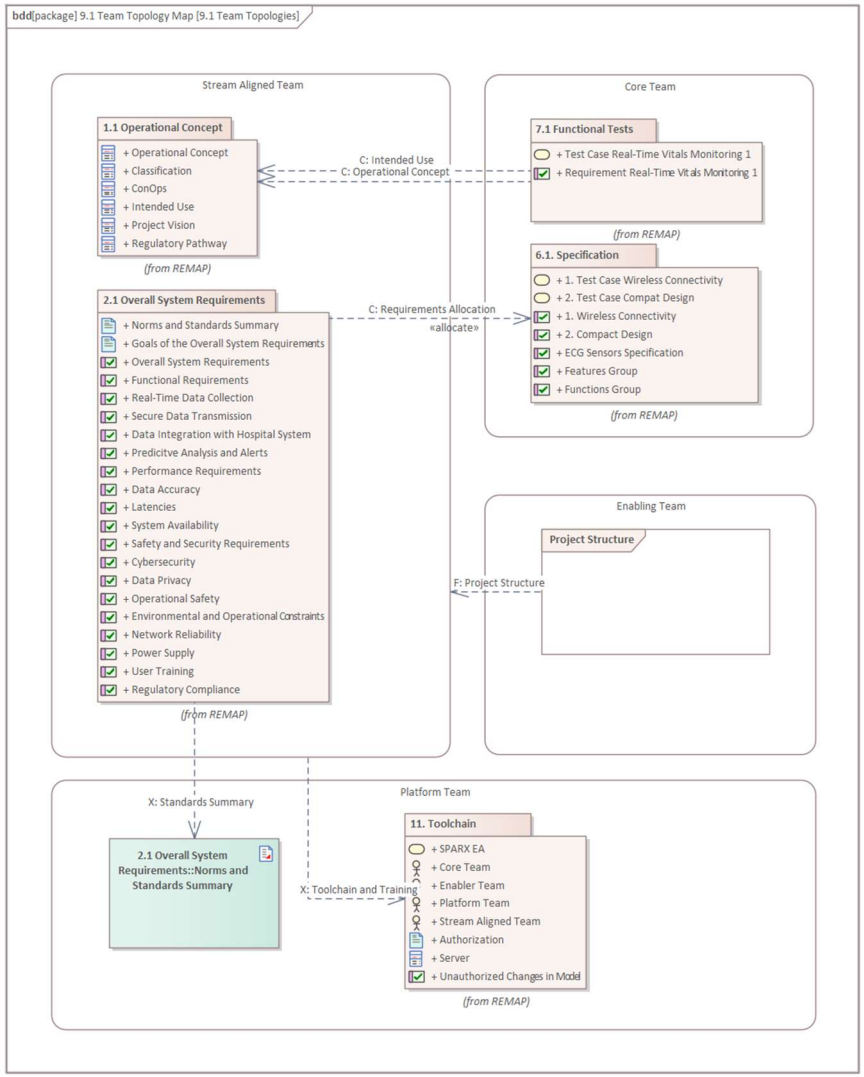


<Company Name> 11 January, 2025

Table of Contents

The HAZOP analysis is a very good way to meet the risk analysis requirements of the Cyber Resilience Act.

Team Topology



The figure shows the **REMAP project's team topology (9.1 Team Topology Map).** It describes the responsibilities of different teams and their contributions to the system's development, verification, and management. The structure promotes collaboration and efficiency in implementing the project goals.

Main components:

1. **Stream Aligned Team:**
    - Focuses on defining and maintaining the **Operational Concept** (1.1) and the **Overall System Requirements** (2.1).

    Main tasks:

    - Development of concepts such as Intended **Use, ConOps, and regulatory paths.**
    - Ensuring compliance with norms and standards and deriving **functional and safety-related requirements.**
2. **Core Team:**
    - Responsible for the implementation and verification of system requirements:
    - **Functional tests (7.1):** Validation of **real-time vital signs monitoring through test cases.**
    - **Specification (6.1):** Creation of detailed specifications for connectivity, design, and functions (e.g., wireless communication and compact design).
3. **Enabling team:**
    - Supports the project structure and the **provision of necessary resources.**
      Task: Coordinating working methods and ensuring cooperation between teams
4. **Platform Team:**
    - Responsible for the provision and maintenance of the **toolchain (11)** and training resources:
    - **Use of tools such as SPARX EA.**
    - **Management of authorization and server processes.**
    - **Ensuring the traceability** of changes and system integrity.

Contexts:

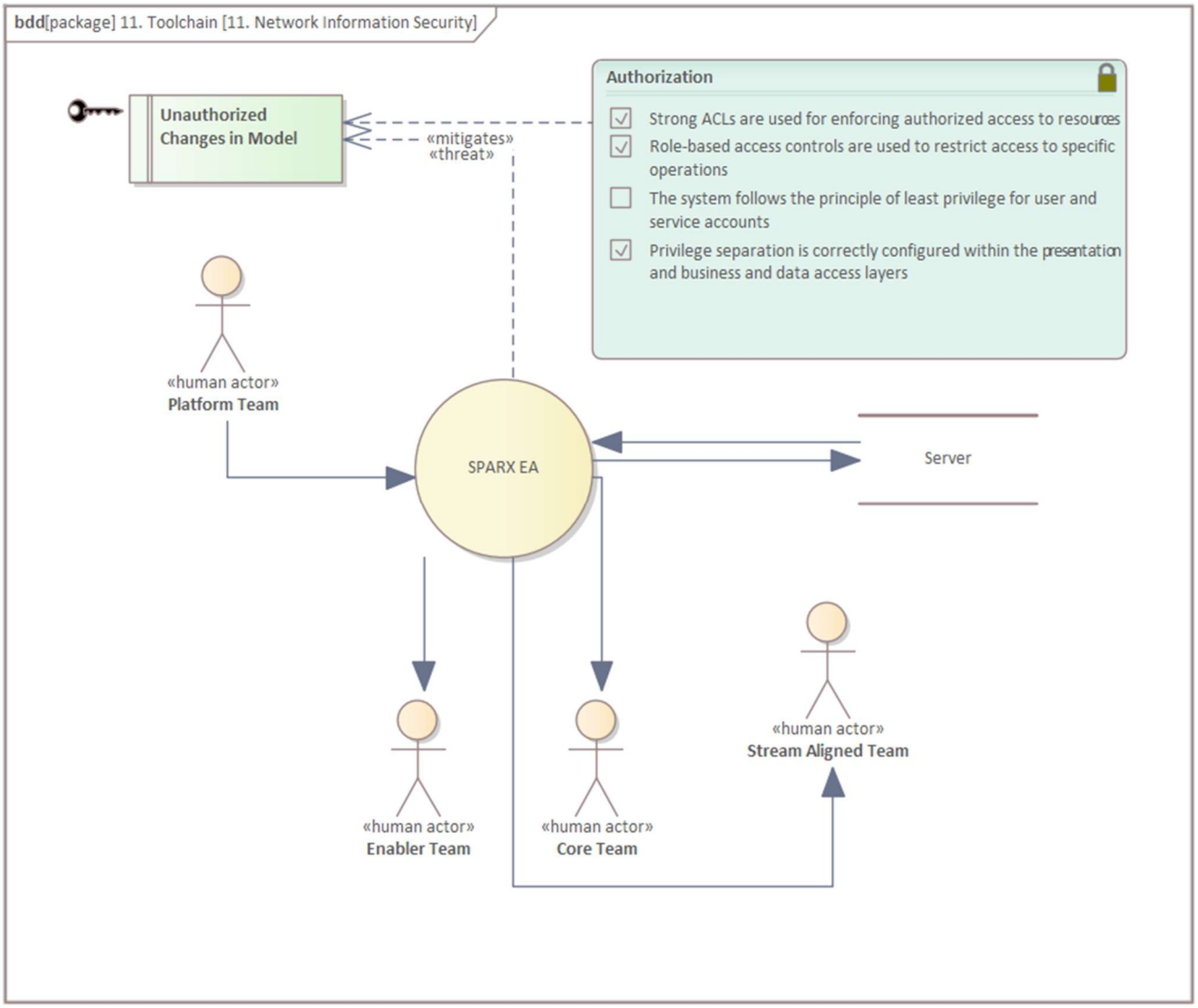
- **The Operational Concept (1.1) forms the basis for the Overall System Requirements (2.1),** which are managed by the Stream-Aligned Team.
- Requirements are forwarded to the core team for implementation and verification.
- **The toolchain (11)** provides central support for all teams to standardize and automate work processes.

- Compliance with standards and norms is ensured by the platform and stream teams.

Aim of the topology:

This structure promotes team specialization and cooperation by defining clear responsibilities and interfaces. It ensures that all aspects of the REMAP project are implemented efficiently and in accordance with standards.

Toolchain Risk Analysis



The figure shows the **REMAP project's toolchain (11), which focuses** on managing network security and preventing unauthorized changes to the model. At the center is the **SPARX EA** tool, which supports the modeling and collaboration processes between the teams.

Main components:

1. **SPARX EA (central modeling tool):**
    - Enables the management, modeling, and traceability of system architectures and requirements.
    - Provides central access points for all teams (Platform Team, Enabler Team, Core Team, Stream Aligned Team).
2. **Teams:**
    - **Platform Team:** Manage the toolchain and provide the system infrastructure.
    - **Enabler Team:** Supports the modeling and development processes by providing resources and technical support to the teams.
    - **Core Team:** Works directly on system implementation and verification.
    - **Stream Aligned Team:** Focused on defining requirements and monitoring the project's progress.
3. **Server:**
    - Ensures the connection between SPARX EA and the teams and enables centralized data storage and access control.
4. **Unauthorized Changes in Model:**
    - Identifies the risk of unauthorized changes in the model.

Mitigations:

Authorization:

- Use strong access control lists (ACLs) to ensure authorized resource access.
- Role-based access controls (RBAC): Restrict access to specific operations.
- Application of the least privilege principle for users and service accounts.
- Correct configuration of **privilege separation** between presentation, business, and data access layers.

Goals of the toolchain:

- **Security:** Protection against unauthorized changes and access.
- **Efficiency:** Centralization of modeling and collaboration.
- **Transparency:** Traceability of changes and assignment of responsibilities.
- **Collaboration:** Support for all teams through a standardized platform.

The figure underlines SPARX EA's central role in promoting collaboration and ensuring the integrity and security of the REMAP project model.

Conclusion

The presented concept, consisting of integrating MBSE, the V-model, and modern team topologies, offers a future-oriented and powerful solution for developing complex, safety-critical cyber-physical systems (CPS). It overcomes the limitations of traditional approaches by focusing on interdisciplinarity, efficiency, and agility.

The combination of these approaches specifically addresses the challenges that arise in the development of CPS, such as the isolation of teams, a lack of traceability, and a lack of flexibility. **Model-based system development (MBSE)** creates a consistent and traceable basis for all development phases, while **team topologies** improve collaboration and break down silo structures. Combined with the proven **V-model**, this creates a framework that provides both the structure and flexibility to manage dynamic requirements and complex system architectures effectively.

The concept makes development more efficient, error-free, and adaptable, which is particularly essential for safety-critical systems such as the REMAP system.

Advantages of the concept:

1. Personnel value add

- **Reduction of cognitive load**: Using team topologies (e.g., stream-aligned, enabler, core, and platform teams) minimizes the cognitive load on teams. Teams can concentrate on their core tasks, which increases employee productivity and satisfaction.
- **Interdisciplinary collaboration**: Breaking down silos through clear interfaces and APIs promotes communication and cooperation between different teams, making projects more efficient and less conflictual.
- **Targeted training and further education**: Enabling teams can support employees in introducing new technologies and methods, such as MBSE, which increases their ability to innovate.

2. Technological value add

- **Efficiency through MBSE**: The model-based system technology offers a "single source of truth," which reduces redundant work and minimizes errors in the documentation. Changes can be automatically adopted in the model, speeding up iterations.
- **Reusable system models**: MBSE enables the reuse of models and components in future projects, significantly reducing development time and costs.
- **Early error detection**: MBSE tools and standardized processes help identify potential weaknesses and risks at an early stage, avoiding expensive rework.
- **Integrated safety analyses**: The early integration of safety analyses (e.g., HAZOP) minimizes risks and increases the reliability of the developed systems.

3. Organizational value add

- **Optimized team structures**: The introduction of modern team topologies increases the organization's efficiency and agility. Teams are flexibly scalable and can be dynamically adapted to project requirements.
- **Reduction of rework**: Clear traceability throughout the development cycle reduces costly corrections and rework phases.

- **Better planning and transparency**: The V-model and MBSE offer structured processes and clear milestones, which enable better project control and planning.
- **Faster time to market**: Optimized processes and clear responsibilities enable companies to shorten development times and bring new products to market faster.

The combination of MBSE, V-model, and team topologies offers companies a strategic opportunity to reduce costs, optimize processes, and increase the quality of their products. This leads to competitive advantages and better fulfillment of customer and regulatory requirements, especially in highly regulated industries such as medicine and other highly innovative areas.

The REMAP system presented here underlines the concept's practical applicability. It processes real-time data from patients in ambulances and ensures secure and efficient integration between the physical and digital components.

In summary, the presented concept optimizes the development of safety-critical CPS by increasing efficiency, improving quality, and offering agile adaptability simultaneously. This makes it particularly valuable for highly regulated industries such as medicine or the automotive and aviation sectors, where precise, safe, and reliable systems are essential.