A logo with blue circles and black text

Description automatically generated

|  |
| --- |
| Technische Hochschule Ulm |
| Digital Twin of Kuka KR3 |
| System Requirements |

|  |
| --- |
| Ibrahim Almohamed, Ahmed  21.08.2024 |

Contents

[1. Version and Control 2](#_Toc176408043)

[2. Glossary 3](#_Toc176408044)

[3. System Overview 5](#_Toc176408045)

[4. System Components 6](#_Toc176408046)

[Dashboard 6](#_Toc176408047)

[OPCUA Server 6](#_Toc176408048)

[ROS2-KR3 Core 6](#_Toc176408049)

[KR3-KUKAVARPROXY 7](#_Toc176408050)

[Database SQL 7](#_Toc176408051)

[Communication Flow 7](#_Toc176408052)

[5. App Sketching 8](#_Toc176408053)

[a. LoginScreen 8](#_Toc176408054)

[b. HomeScreen 9](#_Toc176408055)

[c. DevicesScreen 10](#_Toc176408056)

[d. ConnectionScreen 12](#_Toc176408057)

[e. SimulationScreen 13](#_Toc176408058)

[6. System Use cases 15](#_Toc176408059)

[7. System Requirements 26](#_Toc176408060)

[a. Functional Requirements 26](#_Toc176408061)

[8. Templates 27](#_Toc176408062)

# Version and Control

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Name of Editor | Notes | Date |
| 1.0.0 | Ahmed Ibrahim Almohamed | Related to CustomerRequirementV1\_0\_0.docx | 21.08.2024 |
|  |  |  |  |
|  |  |  |  |

# Glossary

|  |  |
| --- | --- |
| Term | Description |
| DT | Digital Twin |
| KukaDigitalTwin | A Digital twin system of the Kuka KR3 using ROS and Gazebo (simulation tool) . |
| KukaVerse | A Dashboard subsystem that creates a OPCUA server and control the DT . |
| ROS2KR3Core | A software based on ROS2 that runs and manages the simulation of the KR3 Digital twin. |
| ROS2KR3Connection | A connection method to connect the ROS2 (or the host PC) with the KUKA KR3 robot using KUKAVARPROXY . |
| ROS2OPCUABridge | a Software bridge to map a ROS2 node to a OPCUA node. |
| AKL | “Automatisches Kleinteilelager” (DE) or “Automated small parts warehouse” (EN) |
| ROS | Robot Operating System |
| Kuka KR3 | A robotic arm with a microscope  Description automatically generated |
| KVP | KUKAVARPROXY |
| OPC-UA |  |
| SoftRealTime | system where deadlines are important but missing them occasionally does not result in system failure.(average delay of 5ms-30ms) |
| BiDirectionConnection | A connection between the physical and digital robots where commands can be sent from either robot to control the other, and the state information (such as position, velocity, sensor data, etc.) is continuously exchanged. |
| MoveIt2 | A robotic manipulation platform for ROS 2 and incorporates the latest advances in motion planning, manipulation, 3D perception, kinematics, control, and navigation |
| RosInterface | A software interface for the Ros2 to connect the Controllers and the simulation of Gazebo with the KVP protocol. |
| GUI | Graphical User Interface |
| RoboticsLab | A Laboratory at the THU that is used for running experiments of robotics. |
| KukaDigitalTwinDashboard | A Dashboard which is a part of the KukadigitalTwin GUI , used for control and monitor the digital twin and the real twin. |
| RosTasks | A RosTask is a software that aims to create a simple or complicated task for the KukaDigitalTwin , where the user writes a RosNode ,that is runnable on both the physical and digital twins. |
| RQT | RQT is a graphical user interface (GUI) tool for ROS 2. Everything done in RQT can be done on the command line, but RQT provides a more user-friendly way to manipulate ROS 2 elements. |
| RosNode | A node is a participant in the ROS 2 graph, which uses a client library to communicate with other nodes. Nodes can communicate with other nodes within the same process, in a different process, or on a different machine. Nodes are typically the unit of computation in a ROS graph; each node should do one logical thing. |

# System Overview

A diagram of a computer server

Description automatically generated

Figure 1: KR3 Digital Twin System Overview

The system architecture depicted in the diagram represents an integrated setup for controlling and monitoring a KUKA KR3 robot using both a real and simulated environment. The architecture leverages the OPC UA (Open Platform Communications Unified Architecture) standard for communication between various components, ensuring a flexible, scalable, and secure system. The primary components involved include a Dashboard, OPCUA Server, ROS2-KR3 Core, KR3-KUKAVARPROXY, and a SQL Database for data management.

The use of OPC UA as a communication standard ensures seamless integration between the different components, while the ROS2 core enables sophisticated simulation capabilities. The architecture goal is to support real-time control, monitoring, and data logging, making it suitable for complex robotic applications where both accuracy and reliability are crucial.

In this Document it will defined the System use cases and the system requirements ,due to the System-Engineering approach of developing this system .

# System Components

## Dashboard

* **Role**: The Dashboard serves as the primary user interface for operators to control and monitor the KR3 robot. Users can issue commands, view real-time data, and monitor the status of both the real and simulated robots.
* **Interaction**: The Dashboard communicates with the system via an OPCUA-Client, sending user commands to the OPCUA Server and receiving feedback and status updates.

## OPCUA Server

* **Role**: The OPCUA Server acts as the central communication hub for the system. It manages the interactions between the Dashboard, the ROS2 simulation environment, and the real KR3 robot through the KUKAVARPROXY.
* **Components**:
  + **OPCUA-JOpenShowVar**: This module within the OPCUA Server is responsible for translating OPC UA commands into TCP commands that the KUKAVARPROXY can understand.
  + **OpenShowVar Client**: This client interfaces with the KUKAVARPROXY, sending and receiving data over a TCP connection to control the real robot.

## ROS2-KR3 Core

* **Role**: The ROS2-KR3 Core is responsible for simulating the robot's operations within a ROS2 (Robot Operating System 2) environment. It allows for testing and validation of robot commands in a virtual setting before they are executed on the real robot.
* **Components**:
  + **KR3-Simulation**: Handles the simulation of the KR3 robot in the Gazebo environment or similar simulation platforms within ROS2.
  + **KR3-Controller**: Manages the control logic for the robot, interpreting commands and generating appropriate control signals.
  + **KR3 ROS2-OPCUA-Bridge**: This bridge facilitates communication between ROS2 and the OPCUA Server. It acts as an OPCUA-Client within the ROS2-KR3 Core, enabling the core to receive commands from the OPCUA Server and send back status updates.

## KR3-KUKAVARPROXY

* **Role**: The KR3-KUKAVARPROXY is the intermediary that directly interfaces with the physical KR3-R540 robot and its controller (KRC4). It translates high-level commands from the OPCUA Server into specific control commands that the KRC4 can execute.
* **Connection**: The KR3-KUKAVARPROXY communicates with the OPCUA Server over a TCP connection, receiving commands to control the robot's movements and actions.

## Database SQL

* **Role**: The SQL Database is responsible for storing user data, system configurations, and logs of all interactions within the system. It provides a reliable way to track system performance, user commands, and robot status over time.
* **Interaction**: The database interacts with both the ROS2-KR3 Core and the Dashboard, storing ROS2 data as well as user and system data for future retrieval and analysis.

## Communication Flow

1. **User Command**: The operator sends a command via the Dashboard (e.g., "move to point").
2. **OPCUA Server**: The command is sent to the OPCUA Server via the OPCUA-Client in the Dashboard.
3. **Command Distribution**:
   * The OPCUA Server forwards the command to the ROS2-KR3 Core, where it is simulated.
   * Simultaneously, the OPCUA Server sends the command to the KR3-KUKAVARPROXY via a TCP connection, which translates and executes the command on the real KR3-R540 robot.
4. **Feedback Loop**: The KR3-KUKAVARPROXY and ROS2-KR3 Core send status updates back to the OPCUA Server, which then relays this information to the Dashboard for user monitoring.
5. **Data Logging**: Throughout the process, relevant data (commands, robot status, simulation data) is logged into the SQL Database for tracking and analysis.

# App Sketching

## LoginScreen

A screenshot of a login form

Description automatically generated

Figure 2: KukaVerse:Login

This is the start point of the KukaVerse , where the use can login or reset the password.

## HomeScreen

A screenshot of a computer

Description automatically generated

Figure 3: KukaVerse:Home

This is the HomeScreen of the KukaVerse , where the user can see some information about the system.

On the side there is also the is the sections of the app , the state of app connection and the username of the current user.

The Simulate button is disabled here until a connection to the robot is established.

## DevicesScreen

A screenshot of a computer

Description automatically generated

Figure 4: KukaVerse: Devices

The DevicesScreen is the main screen where all the current users robots and their configurations are displayed .

The user can also make new entries here by clicking on the “+” icon in the bottom right corner

.

Which leads to the AddDeviceScreen

A screenshot of a computer

Description automatically generated

Figure 5: KukaVerse : Add Device

On this screen the user can create a new robot with a description , type and name (and more configurations are TBD).

The user saves the new robot so it will be displayed as a card on the DevicesScreen.

## ConnectionScreen

A screenshot of a computer

Description automatically generated

Figure 6: KukaVerse : Connection

On the ConnectionScreen the user can configure the connection to the OPCUA server ,that starts when the connect button is pressed.

The user can set a Device( from stored devices) ,an Endpoint URL and a logging level.

After a connection is established the State “Not connected” changes to “Connected” and the simulate button is unlocked.

## SimulationScreen

A screenshot of a computer

Description automatically generated

Figure 7: KukaVerse : Simulate

On the SimulationScreen , The user views the device and connection configurations.

After that the user Starts the Simulation by pressing the start simulation button that starts the Gazebo on another window and starts the KukaVerse control panel in single mode.

A screenshot of a computer

Description automatically generated

Figure 8: KukaVerse : Control panel :Single

The User here can set a point (x,y,z) and a speed and click on the Go to Point button (the robot here shall start moving to that point and also the simulation).

The user can also check the Set Multi-Points option to save a set of point the robot reach (point1 -> point2 -> point3 ….) after setting each point by clicking the set point Button , and starting a robot job with go to point.

A screenshot of a computer

Description automatically generated

Figure 9: KukaVerse : Control panel Multi

# System Use cases

A diagram of a computer

Description automatically generated

Figure 10: System Use cases

|  |  |
| --- | --- |
| **Identifier** | **UC1** |
| **Name** | LoginMode |
| **Description** | The KukaVerse logs the user in |
| **Preconditions** | The KukaVerse is started |
| **Postconditions** | The KukaVerse is starts on UC2:HomeMode  OR  The KukaVerse is on UC1:LoginMode |
| **Failure Scenarios** | n/a |
| **Related Usecases** | n/a |
| **Actors** | User |
| **Trigger** | The User starts the KukaVerse app |
| **Standard Procedure** | The KukaVerse starts the LoginScreen  The User inputs the UserEmail and UserPassword  The User presses the LoginButton  The KukaVerse displays a “LoginSuccessfulMessage” for 3 seconds and continues with UC2: HomeMode |
| **Alternative Procedure** | 4a.1 The KukaVerse displays “LoginFaildMessage” for 3 seconds    4a.2 The KukaVerse continues with UC1:LoginMode |

|  |  |
| --- | --- |
| **Identifier** | **UC2** |
| **Name** | HomeMode |
| **Description** | The KukaVerse displays the HomeScreen and the MainDashboard |
| **Preconditions** | The User logs in successfully |
| **Postconditions** | The KukaVerse is in UC3: DevicesMode  OR  The KukaVerse is in UC5:ConnectionMode |
| **Failure Scenarios** | n/a |
| **Related Usecases** | n/a |
| **Actors** | User |
| **Trigger** | The User logs in successfully |
| **Standard Procedure** | 1. The KukaVerse displays KukaVerseInfo 2. The KukaVerse displays the list of KukaVerseFunctionalityList and “SystemNotConnectedMessage” 3. The User selects “Devices” form the KukaVerseFunctionalityList then the KukaVerse continues with UC3:DevicesMode |
| **Alternative Procedure** | 3a The User selects “Connect” from the KukaVerseFunctionalityList then the KukaVerse continues with UC4:ConnectionMode |

|  |  |
| --- | --- |
| **Identifier** | **UC3** |
| **Name** | DevicesMode |
| **Description** | The KukaVerse displays the DevicesList |
| **Preconditions** | The KukaVerse is in UC2:HomeMode  OR  The KukaVerse is in UC5:ConnectionMode |
| **Postconditions** | The KukaVerse is in UC2: HomeMode  OR  The KukaVerse is in UC4:AddDeviceMode  OR  The KukaVerse is in UC5:ConnectionMode |
| **Failure Scenarios** | n/a |
| **Related Usecases** | n/a |
| **Actors** | User |
| **Trigger** | The User selects “Devices” from the KukaVerseFunctionalityList |
| **Standard Procedure** | 1. The KukaVerse displays all stored DeviceCard on DevicesScreen 2. The KukaVerse displays the list of KukaVerseFunctionalityList and “SystemNotConnectedMessage” 3. The KukaVerse displays AddDeviceButton 4. If the User pushes the AddDeviceButton then the KukaVerse continues with UC4:AddDeviceMode |
| **Alternative Procedure** | 1a. If there is no DeviceCard stored then the KukaVerse displays DefaultDeviceCard  4a. If the User selects “Connect” from the KukaVerseFunctionalityList then the KukaVerse continues with UC4:ConnectionMode  4a. If the User “Home” from the KukaVerseFunctionalityList then the KukaVerse continues with UC2:HomeMode |

|  |  |
| --- | --- |
| **Identifier** | **UC4** |
| **Name** | AddDeviceMode |
| **Description** | The User adds and stores a new DeviceCard . |
| **Preconditions** | The KukaVerse is in UC3:DevicesMode |
| **Postconditions** | The KukaVerse is in UC3:DevicesMode |
| **Failure Scenarios** | n/a |
| **Related Usecases** | n/a |
| **Actors** | User |
| **Trigger** | The User pushes The User pushes the AddDeviceButton on the DevicesScreen |
| **Standard Procedure** | 1. The KukaVerse displays the NewDevicePannel on AddDeviceScreen 2. The User inputs a NewDeviceName 3. The User selects a NewDeviceType 4. The User inputs a NewDeviceDescription 5. If the User selects SaveDeviceButton then the KukaVerse stores the NewDevice and continues with UC3: DevicesMode with a new DeviceCard on the DevicesScreen. |
| **Alternative Procedure** | 5a. If the User selects CancelButton then the KukaVerse continues with UC3:DevicesMode with no NewDevice and no new DeviceCard on the DevicesScreem |

|  |  |
| --- | --- |
| **Identifier** | **UC5** |
| **Name** | ConnectionMode |
| **Description** | The KukaVerse displays the ConnectionPannl on the ConnectionScreen |
| **Preconditions** | The KukaVerse is in UC2:HomeMode  OR  The KukaVerse is in UC3:DevicesMode |
| **Postconditions** | The KukaVerse is in UC6:SimluationMode  OR  The KukaVerse is in UC3:DevicesMode  OR  The KukaVerse is in UC2:HomeMode |
| **Failure Scenarios** | KukaVerse fails to connect the KukaVerseOpcuaClient to the KukaOpcuaServer. |
| **Related Usecases** | n/a |
| **Actors** | User |
| **Trigger** | The User selects “Devices” from the KukaVerseFunctionalityList |
| **Standard Procedure** | 1. The KukaVerse displays the ConnectionScreen with ConnectionPannel 2. The User selects the Device to be connected 3. The User inputs the EndpointURL 4. The User selects the LoggingLevel of the connection 5. If The User pushes the ConnectButton then the KukaVerse start the following:    1. Connects the KukaVerseOpcuaClient to the KukaOpcuaServer    2. Displays a “SystemConnectedMessage”    3. Enables the “Simulate” option on the KukaVerseFunctionalityList 6. If the User selects “Simulate” from the KukaVerseFunctionalityList then the KukaVerse continues with UC6:SimulationMode |
| **Alternative Procedure** | 5a if KukaVerse fails to connect the KukaVerseOpcuaClient to the KukaOpcuaServer then the KukaVerse starts the following :   * 1. displays “OpcuaConnectionErrorMessage”   2. displays “SystemNotConnectedMessage”   3. disables the “Simulate” option on the KukaVerseFunctionalityList if it was enabled   6a If the User “Home” from the KukaVerseFunctionalityList then the KukaVerse continues with UC2:HomeMode  6a if the User selects “Devices” form the KukaVerseFunctionalityList then the KukaVerse continues with UC3:DevicesMode |

|  |  |
| --- | --- |
| **Identifier** | **UC6** |
| **Name** | SimulationMode |
| **Description** | The KukaVerse displays the selected DeviceCard and the ConnecctionCard |
| **Preconditions** | The KukaVerse displays SystemConnectedMessage |
| **Postconditions** | The KukaVerse is in UC7:ControlPanelMode  OR  The KukaVerse is in UC5:ConnectionMode  OR  The KukaVerse is in UC:3DevicesMode  OR  The KukaVerse is in UC2:HomeMode |
| **Failure Scenarios** | n/a |
| **Related Usecases** | n/a |
| **Actors** | User |
| **Trigger** | The User selects “Simulate” from the KukaVerseFunctionalityList |
| **Standard Procedure** | 1. The KukaVerse displays the following :    1. DeviceCard about the selected Device    2. ConnectionCard about the configured connection. 2. If the User pushes the StartSimulationButton then the KukaVerse continues with UC7:ControlPanelMode |
| **Alternative Procedure** | 2a if the User selects “Devices” form the KukaVerseFunctionalityList then the KukaVerse continues with UC3:DevicesMode  2a if the User selects “Connect” form the KukaVerseFunctionalityList then the KukaVerse continues with UC5: ConnectionMode  2a if the User selects “Home” form the KukaVerseFunctionalityList then the KukaVerse continues with UC2:HomeMode |

|  |  |
| --- | --- |
| **Identifier** | **UC7** |
| **Name** | ControlPanelMode |
| **Description** | The KukaVerse displays ControlPannel on the ControlPannelScreen |
| **Preconditions** | The user pushes the “StartSimualtionButton” in UC6:SimulationMode |
| **Postconditions** | The KukaVerse is in UC7: ControlPanelMode |
| **Failure Scenarios** | n/a |
| **Related Usecases** | n/a |
| **Actors** | User |
| **Trigger** | The User pushes the StartSimulationButton on SimulationScreen in UC6:SimulationMode |
| **Standard Procedure** | 1. The KukaVerse displays the ControlPannel 2. The User sets the following of the PointPannel:    1. The RobotSpeed    2. The GoToPointXCoordinate    3. The GoToPointYCoordinate    4. The GoToPointZCoordinate 3. The User pushes the GoToPointButton 4. The KukaVerse sends GoToPointCommand to the KukaOpcuaServer 5. The KukaVerse displays the following :    1. The RobotStateCard    2. The SimulationStateCard    3. A DefaultPointsCard 6. The KukaVerse continues with UC7: ControlPanelMode |
| **Alternative Procedure** | 1a. The User checks the SetMultiPointsOption  3a. The User pushes the SetPointButton and continues on step 2 of the Standard Prorcedure (steps 1a and 3a repeated for maximum of 4 points)  5a. The KukaVerse displays the following :   * 1. The RobotStateCard   2. The SimulationStateCard   3. The PointToReachCard |

|  |  |
| --- | --- |
| **Identifier** | **UC8** |
| **Name** | LogoutMode |
| **Description** | The KukaVerse logs the user out |
| **Preconditions** | The KukaVerse is in any state |
| **Postconditions** | The KukaVerse is in UC1:LoginMode |
| **Failure Scenarios** | n/a |
| **Related Usecases** | n/a |
| **Actors** | User |
| **Trigger** | The User selects the “Logout” option on the KukaVerseFunctionalityList |
| **Standard Procedure** | 1. The KukaVerse stops any running simulations 2. The KukaVerse disconnects the KukaVerseOpcuaClient from the KukaOpcuaServer 3. The KukaVerse dispalys “LogoutMessage” 4. The KukaVerse continues with UC2:LoginMode |
| **Alternative Procedure** | n/a |

# System Requirements

## Functional Requirements

# Templates

|  |  |
| --- | --- |
| Requirement ID |  |
|  |  |

|  |  |
| --- | --- |
| **Identifier** |  |
| **Name** |  |
| **Description** |  |
| **Preconditions** |  |
| **Postconditions** |  |
| **Failure Scenarios** |  |
| **Related Usecases** |  |
| **Actors** |  |
| **Trigger** |  |
| **Standard Procedure** |  |
| **Alternative Procedure** |  |