marek.tatara@dac.digital

Document title: Use cases analysis

Version	Status	Date
2.1	final	2020-12-03

Author
F. Montori, M. Tatara

Contact
federico.montori2@unibo.it

Use cases analysis

DAC: Marek Tatara
marek.tatara@dac.digital

IUNET: Federico Montori
federico.montori2@unibo.it

Abstract

In this document, we analyze each use case starting from their architecture. We then address their gaps by proposing dedicated needed tools and toolchains for each of them. For each tool, following the approach and the best practices stated in O2, we outline their main characteristics that are necessary to frame them within the big picture of its use case as well as in the phases of the engineering process proposed.

ECSEL Joint Undertaking

ECSEL EU project 826452 - Arrowhead Tools
Project Coordinator: Professor Jerker Delsing | Luleå University of Technology

Table of contents

In this document we will perform an analysis over all use cases with respect to their toolchain architecture, i.e., what tools are used at the beginning of the projects, which new tools are going to be developed and which tools are going to be improved. We will also provide insights on what engineering process phases (EPPs) are covered by such tools.
In particular, for each use case, we will present: the status of the toolchain architecture in the baseline (action done and tools used), then we will proceed with a gap analysis to understand which shortcomings are encountered and, finally, which new tools (or modifications of the old ones) are to fill such gaps.

1. UC-01 [CAMEA, BUT, CVUT, EXPLEO and AIT]: Automated Formal Verification

Contact: Lukáš Maršík [l.marsik@camea.cz]
The description of each Use Case can be found in D1.2

1.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 1

AHT-EPP	**Smart camera system**
1 Requirements	Requirements are obtained from past experience of CAMEA experts and from discussions between CAMEA representatives and customers. They are tracked using Word and Excel documents, sometimes even staying in the memory of CAMEA experts who then reflect them in the system design, code assertions, and the tests being manually prepared.
2 Functional Design	Simple block models representing prototypes of the main functional components are developed using languages like Python, C#, or C++. Sometimes Matlab is used (in particular, when dealing with AI-based components). Repeated test or simulation runs are manually performed to assess properties of the developed prototypes.
3 Procurement & Engineering	Based on experience gained from prototypes (provided they are built), final versions of the systems under development are written in C#, C++, or C (the latter is used in particular for drivers). Sometimes, prototypes developed in C# are just further optimised and used as the final product. In case of hardware-accelerated processing, VHDL code is produced, and synthesis for FPGA is subsequently used. For developing GUI, Visual Studio and C# are used. Unit tests and system tests are used in a manual way to check whether the system produced satisfies the collected requirements. Typing and basic static analysis checks built into Visual Studio and CLion are used. For the HW developed, manual comparisons of its behaviour with the behaviour of a C#/C++ prototype are used.
4 Deployment & Commissioning	Not relevant.
5 Operation & Management	Not relevant.
6 Maintenance Decommissioning & Recycling	Not relevant.
7 Evolution	Not relevant.
8 Training & Education	No training in advanced analysis, verification, or testing approaches.

1.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 1

AHT-EPP	**Smart camera system**
1 Requirements	Microsoft Word and Excel or other similar text or spreadsheet processors.
2 Functional Design	Python, C#, C++, Matlab, Visual Studio, CLion.
3 Procurement & Engineering	C#, C++, C, GNU tool suite, Visual Studio, CLion, Vivado.
4 Deployment & Commissioning	Not relevant.
5 Operation & Management	Not relevant.
6 Maintenance Decommissioning & Recycling	Not relevant.
7 Evolution	Not relevant.
8 Training & Education	No training in advanced analysis, verification, or testing approaches.

1.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 1

AHT-EPP	**Smart camera system**
1 Requirements	Manual.
2 Functional Design	Manual, up to automation of the build process in the IDEs used.

3 Procurement & Engineering	Manual up to automation of the build process in the IDEs used (Visual Studio, CLion) and up to the automation provided by the IDEs for running manually prepared tests and for typing and basic static analysis built into the IDEs.
4 Deployment & Commissioning	Not relevant.
5 Operation & Management	Not relevant.
6 Maintenance Decommissioning & Recycling	Not relevant.
7 Evolution	Not relevant.
8 Training & Education	No training in advanced analysis, verification, or testing approaches.

1.4. Gap Analysis

1.4.1. Level of automation of the toolchain & Information passed

The level of the automation of the baseline is restricted to the automation provided by the IDEs used (Visual Studio, CLion), i.e., to an automated build process, automatic running of built-in typing and basic static analysis checks, and automated running of manually prepared tests.

Identified requirements are transferred manually to the phases of functional design as well as procurement and engineering. Knowledge gained by functional design is transferred manually to engineering, unless the prototype code built within functional design is used as a basis and gradually extended and/or optimized in the phase of engineering.

1.4.2. Level of integration with the Arrowhead Framework

The toolchain is not integrated into the framework yet. We are currently working on that as the next immediate step.

1.4.3. Missing Tools within the UC

Use Case 1 aims at applications of advanced automated analysis, verification, and testing methods with formal roots within the design of embedded systems with a stress on the smart cameras being developed by CAMEA (and their underlying technologies). These methods may be aided by modeling techniques wherever appropriate too. Tools that are currently missing from this point of view and that the partners plan to provide are the following:

- Model-based tools: Gamma Statechart Composition Framework, Uppaal for timed analysis, PRISM for reliability analysis, Modica, Testona.
- Advanced static analysis tools: various analysis plugins for platforms such as Facebook Infer or Frama-C.

- Advanced testing and dynamic analysis tools: ANaConDA, Perun, Testos.
- Sound formal analysis and verification: 2LS, Predator/Symbiotic.

Since the UC should consider a variety of different approaches to analysis and verification, at least one representative of the different considered approaches should be used (i.e., at least one static analyzer, one dynamic analyzer, one formal verifier, and one model-based tool).

1.4.4. Missing Tools Outside the UC

The range of tools mentioned in the previous point is sufficient for the UC. However, further advanced analysis, verification, testing, or synthesis tools can be added if they are offered by project partners.

1.5. Declared Tools

Development of completely new tools is not excluded, but stress will be put on improvements of already existing tools being developed or co-developed by the involved partners, e.g., (e.g., ANaConDA, Predator, 2LS, Testos, Storm, Perun, Gamma Statechart Composition Framework, Modica, Testona, Facebook Infer, Frama-C, CPAchecker, Symbiotic, etc.).

Name	This is a tool in the following phase	Compatible with AHF	Inputs	Outputs
Smart camera system	When restricted to the AHT connector of the smart cameras, the development has not yet started.	Once developed, the connector will be compatible with AHF: indeed, this is the purpose why it is to be developed.	Control and configuration commands for smart camera device	State information of camera detection cores, system information (mostly OS-related), HW related information (e.g. temperature, voltage, power consumption, ...)
Smart camera AHF connector	The development has not yet started.	Once developed, the connector will be compatible with AHF: indeed, this is the purpose why it is to be developed.	Control and configuration commands for smart camera device coming from AHF	State information of camera detection cores, system information (mostly OS-related), HW related information (e.g. temperature, voltage, power consumption, ...) sent to AHF
Advanced analysis, Verification, and Testing Tools	First prototypes of several advanced analysis, verification and testing tools potentially usable in the UC exist but have not yet been applied there.	These are development tools and so they will not be directly connected to the AHF core. They will be applied to verify some systems used in connection with AHF or even some subsystems or underlying technologies of AHF itself.	Source code, binary code, specification of properties to be analyzed.	Potential errors, diagnostic information.

2. UC-02.1 [PHC, TUE] Engineering processes and tool chains for digitalized and networked diagnostic imaging

Contact: Önder Babur, Mahdi Saeedi Nikoo [o.babur@tue.nl m.saeedi.nikoo@tue.nl]
The description of each Use Case can be found in D1.2

This use case was not present in D4.1, the description is fresh.

2.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 3

AHT-EPP	IBM Rhapsody	Matlab/Simulink	Unity
1 Requirements	Requirements w.r.t. to the previous project at TU/e: the toolchain and the functionality of each tool as well as the nature of interaction will remain the same. New requirements w.r.t. to the framework: the communication and data exchange should be streamlined over the Arrowhead-Framework.	Requirements w.r.t. to the previous project at TU/e: the toolchain and the functionality of each tool as well as the nature of interaction will remain the same. New requirements w.r.t. to the framework: the communication and data exchange should be streamlined over the Arrowhead-Framework.	Requirements w.r.t. to the previous project at TU/e: the toolchain and the functionality of each tool as well as the nature of interaction will remain the same. New requirements w.r.t. to the framework: the communication and data exchange should be streamlined over the Arrowhead-Framework.
2 Functional Design	SysML profile and modeling for the framework, core systems as well as the toolchain in this use case.	Mechanical design/modeling for specific parts of the truck, as used in the previous project.	Modeling of the 3d environment, as well as optionally additional functionality such as collision detection in the future iterations.
3 Procurement & Engineering	Code generation from the profile and integration of the generated code from the profiles with the legacy code generated by IBM Rhapsody (e.g. for integration with Simulink)	Code generation from the profile and integration of the generated code from the profiles with the legacy code generated by IBM Rhapsody (e.g. for integration with Simulink)	Code generation from the profile and integration of the generated code from the profiles with the legacy code generated by IBM Rhapsody (e.g. for integration with Simulink)
4 Deployment & Commissioning	Packaging and compilation of the full tool chain.	Packaging and compilation of the full tool chain.	Packaging and compilation of the full tool chain.
5 Operation & Management	Operation in the context of the Master thesis to be demonstrated.	Operation in the context of the Master thesis to be demonstrated.	Operation in the context of the Master thesis to be demonstrated.
6 Maintenance Decommissioning & Recycling	-	-	-

7 Evolution	Bugfixing and improvement of features in the coming iterations.	No evolution planned, we will keep the basic components as is.	Bugfixing and improvement of features in the coming iterations.
8 Training & Education	Transfer of user and developer documentation to be considered in the next iterations for training.	Transfer of user and developer documentation to be considered in the next phases for training.	Transfer of user and developer documentation to be considered in the next phases for training.

2.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 3
No tools used in phases 4-8

AHT-EPP	**IBM Rhapsody**	**Matlab/Simulink**	**Unity**
1 Requirements	Microsoft Word/Latex/Adobe PDF to keep track the old and new requirements	Microsoft Word/Latex/Adobe PDF to keep track the old and new requirements	Microsoft Word/Latex/Adobe PDF to keep track the old and new requirements
2 Functional Design	IBM Rhapsody, plus possibly Cameo Systems Modeler if necessary	Matlab/Simulink	Unity
3 Procurement & Engineering	IBM Rhapsody as the IDE, with code generation components possibly from Cameo Systems Modeler	Matlab/Simulink as the IDE	Unity as the IDE

2.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 3
No tools used in phases 4-8

AHT-EPP	**IBM Rhapsody**	**Matlab/Simulink**	**Unity**
1 Requirements	manual connection	manual connection	manual connection
2 Functional Design	automated connection with Matlab/Simulink and Unity	automated connection with IBM Rhapsody	automated connection with IBM Rhapsody
3 Procurement & Engineering	manual design through the IDE, automated code generation,	manual design through the IDE	manual design/coding in the IDE

	semi-automated integration of code		

2.4. Gap Analysis

2.4.1. Level of automation of the toolchain & Information passed

Currently the communication between the legacy tools is done in an automated way, but that is not through the Arrowhead Framework. We plan to have a full automation of the toolchain after the integration is done. We are using metamodels, models and code generation in our workflow, so the information passed from design to code and deployment is managed automatically. However, the rest of the phases (e.g. requirements, evolution) are disjoint from this automated workflow and are to be handled manually through documentation.

2.4.2. Level of integration with the Arrowhead Framework

The toolchain is not integrated into the framework yet. We are currently working on that as the next immediate step.

2.4.3. Missing Tools within the UC

There is no missing tool in our current toolchain, only integration work will be performed.
All tools are mandatory as they each add important functionality to the toolchain. Still we can identify Unity as less important than the other tools (IBM Rhapsody and Matlab Simulink) since it is mainly considered for visualization purposes.

2.4.4. Missing Tools Outside the UC

N/A

2.5. Declared Tools

IBM Rhapsody, Matlab/Simulink and Unity will be integrated to operate over the Arrowhead framework.

Name	This is a tool in the following phase	Compatible with AHF	Inputs	Outputs
IBM Rhapsody	EPP1, EPP2, EPP3	The tool itself is not compatible. The SysML profile will generate an Arrowhead-compliant system	The tool provides model parameters for Unity for visualization.	The tool provides model kinematics as output to Matlab Simulink.

Matlab/Simulink	EPP1, EPP2, EPP3	The tool itself is not compatible. The SysML profile will generate an Arrowhead-compliant system	The tool gets model kinematics as input from IBM Rhapsody.	The tool may provide input to IBM Rhapsody
Unity	EPP1, EPP2, EPP3	The tool itself is not compatible. The SysML profile will generate an Arrowhead-compliant system	The tool will get environment parameters from IBM Rhapsody. The tool will produce visual output on the screen.	The tool will produce visual output on the screen.

3. UC-02.2 [PHC, TUE] Engineering processes and tool chains for digitalized and networked diagnostic imaging

Contact: Önder Babur, Mahdi Saeedi Nikoo [o.babur@tue.nl m.saeedi.nikoo@tue.nl]
The description of each Use Case can be found in D1.2
This use case was not present in D4.1, the description is fresh.

3.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 3

AHT-EPP	Model Repository	Model Analytics & Management	Dashboard
1 Requirements	Requirements gathered by analyzing the operation and needs of the existing MMA prototypes	Requirements gathered by analyzing the operation and needs of the existing MMA prototypes	Requirements gathered by analyzing the operation and needs of the existing MMA prototypes
2 Functional Design	Box-and-arrow design of the workflow.	Box-and-arrow design of the workflow.	Box-and-arrow design of the workflow.
3 Procurement & Engineering	Coding a simple model repository from scratch, natively as a system in the Arrowhead framework	Migration and adaptation of the existing MMA frameworks, to operate natively as systems in the Arrowhead framework	Implementing an adapter to communicate with the Kibana dashboard system
4 Deployment & Commissioning	Packaging and compilation of the full tool chain.	Packaging and compilation of the full tool chain.	Packaging and compilation of the full tool chain.
5 Operation & Management	Operation in close link to TUE-SET research areas will be demonstrated.	Operation in close link to TUE-SET research areas will be demonstrated.	Operation in close link to TUE-SET research areas will be demonstrated.
6 Maintenance Decommissioning & Recycling	-	-	-
7 Evolution	Potentially adding CotS model repositories with advanced functionalities.	Bugfixing and adding of new MMA systems in the coming iterations.	No significant evolution planned.
8 Training & Education	Transfer of user and developer documentation to be considered in the next iterations for training.	Transfer of user and developer documentation to be considered in the next phases for training.	Transfer of user and developer documentation to be considered in the next phases for training.

3.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 3
No tools used in phases 4-8. There is a potential for adding CotS model repositories in the Evolution Phase

AHT-EPP	**Model Repository**	**Model Analytics & Management**	**Dashboard**
1 Requirements	Microsoft Word/Latex/Adobe PDF to keep track of the old and new requirements	Microsoft Word/Latex/Adobe PDF to keep track of the old and new requirements	Microsoft Word/Latex/Adobe PDF to keep track of the old and new requirements
2 Functional Design	Drawing tools for box-and-arrow design	Drawing tools for box-and-arrow design	Drawing tools for box-and-arrow design
3 Procurement & Engineering	Java-based implementation from scratch.	Java-based adaptation for the MMA tools	Java adapter to communicate with Kibana

3.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 3
No tools used in phases 4-8. There is a potential for adding CotS model repositories in the Evolution Phase.

AHT-EPP	**Model Repository**	**Model Analytics & Management**	**Dashboard**
1 Requirements	manual connection	manual connection	manual connection
2 Functional Design	manual connection	manual connection	manual connection
3 Procurement & Engineering	manual coding from scratch	manual coding for the migration	manual coding for the adapter

3.4. Gap Analysis

3.4.1. Level of automation of the toolchain & Information passed

Currently the legacy tools are implemented independently and in an ad-hoc manner. We plan to have a full automation of the toolchain with streamlined systems after the integration is done.

The information flow among phases is mostly manual (through documentation) in this use case.

3.4.2. Level of integration with the Arrowhead Framework

By Milestone 4, part of the tools will be integrated to fully operate with the framework, so that we have a fully functional toolchain.

3.4.3. Missing Tools within the UC

We are implementing the model repository system from scratch, as well as integrating the dashboard; these are the two tools missing from our toolchain.
All tools are mandatory as they each add important functionality to the toolchain. The MMA tools, though offering different functionalities, are interchangeable from the point of the workflow. We will iteratively integrate more of them in the long run.

3.4.4. Missing Tools Outside the UC

N/A

3.5. Declared Tools

Model repository, MMA prototypes and Dashboard system (Kibana) will be integrated to operate over the Arrowhead framework.

Name	This is a tool in the following phase	Compatible with AHF	Inputs	Outputs
Model Repository	EPP1, EPP2, EPP3	We are creating the tool from scratch to be Arrowhead-compliant.	Set of models and model metadata (write)	Set of models and model metadata (read)
Model Analytics & Management	EPP1, EPP2, EPP3	The tools themselves are not compatible. We are adapting them to natively work with the framework.	Set of models and model metadata	Analysis results
Dashboard	EPP1, EPP2, EPP3	The tool itself is not compatible. We are writing an adapter to communicate with the tool.	User input (e.g. analysis requests)	Visualization of analysis results

4. UC-02.3 [PHC, TUE] Engineering processes and tool chains for digitalized and networked diagnostic imaging

Contact: Frans Rosbak, Peter van Der Muelen [frans.rosbak_1@philips.com, peter.van.der.meulen@philips.com]
The description of each Use Case can be found in D1.2
This use case was not present in D4.1, the description is fresh.

No input for this use case, still in the definition phase.

5. UC-03 [ULMA, UC3M, REUSE]: Integration of electronic design automation tools with product lifecycle tools

Contact: Jose María Alvarez Rodríguez [josemaria.alvarez@uc3m.es]
The description of each Use Case can be found in D1.2

5.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 6
Phases 5,6,7 are absent

AHT-EPP	IBM Doors	Requirement Authoring Tool	IBM Rhapsody	Altium Designer	Verification Studio	Knowledge manager
1 Requirements	Requirement definition for the digital hardware under the test. Import System Requirements Specification from EUROPEAN INTEGRATED RAILWAY RADIO ENHANCED NETWORK to Doors		Definition in SysML model a PBS based on EUROPEAN INTEGRATED RAILWAY RADIO ENHANCED NETWORK	Requirement definition of altium adapters to be OSLC compliant		Definition of a PBS based on EUROPEAN INTEGRATED RAILWAY RADIO ENHANCED NETWORK
2 Functional Design			Architectural definition for the digital hardware under the test			Vocabulary domain definition for Altium Designer components
3 Procurement & Engineering				Digital hardware creation (PCB & Schematics) Altium adapter implementation to make altium compatible with OSLC RM		Implement a parser to read BOM of hardware models created in Altium Designer

4 Deployment & Commissioning	Provider & Consumer deployment	Provider & Consumer deployment	Provider & Consumer deployment		Provider & Consumer deployment	Provider & Consumer deployment
8 Training & Education	Altium Adapter demo	SRL Demo		Altium Adapter demo		

5.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 6
Phases 5,6,7 are absent

AHT-EPP	**IBM Doors**	**Requirement Tool**	**IBM Rhapsody**	**Altium Designer**	**Verification Studio**	**Knowledge manager**
1 Requirements	VMWare Workstation: Development virtual machine execution. IBM Doors: Requirement definition Tortoise svn: Backup	MySQL: Database HyperV: Virtual Machine Execution		IBM Doors: Requirement definition	MySQL: Database HyperV: Virtual Machine Execution	MySQL: Database HyperV: Virtual Machine Execution
2 Functional Design						
3 Procurement & Engineering				Eclipse: Provider Implementation. Eclipse: Consumer Implementation. Visual Studio: Provider Implementation. Visual Studio: Consumer Implementation. VMWare Workstation: Development		Knowledge Manager: Patterns management for represent the grammatical structure of requirement, creation of domain vocabulary

				virtual machine execution. Altium Designer: Digital Hardware design implementation . Tortoise svn: Backup		
4 Deployment & Commissioning	VMWare Workstation: Development virtual machine execution. IBM Doors: Doors installation. Doors Web Access:	MySQL: Installation and configuration HyperV: Development virtual machine execution			MySQL: Installation and configuration HyperV: Development virtual machine execution	MySQL: Installation and configuration HyperV: Development virtual machine execution
8 Training & Education	Microsoft Word + Powerpoint: Internal training material. Tortoise svn: Backup Acrobat Reader: User manuals and installation manuals are generated in PDF format.		Camtasia Studio 8: Record a video demo	Microsoft Word + Powerpoint: Internal training material. Acrobat Reader: User manuals and installation manuals are generated in PDF format. Tortoise svn: Backup		Microsoft Word + Powerpoint: Internal training material. Google Drive: Work repository

5.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 6
Phases 5,6,7 are absent

AHT-EPP	**IBM Doors**	**Requirement Tool**	**IBM Rhapsody**	**Altium Designer**	**Verification Studio**	**Knowledge manager**
1 Requirements	VMWare Workstation: Manual. IBM Doors: Requirement definition	MySQL: Manual HyperV: Manual		IBM Doors: Manual	MySQL: Manual HyperV: Manual	MySQL: Manual HyperV: Manual

	Tortoise svn: Manual					
2 Functional Design						
3 Procurement & Engineering				Eclipse: Manual Visual Studio: Manual. VMWare Workstation: Manual Altium Designer: Manual Tortoise svn: Manual		Knowledge Manager: Semiautomatic Vocabulary (Manual) Patterns (Automatic)
4 Deployment & Commissioning	VMWare Workstation: Manual IBM Doors: Installation for our toolchain. Doors Web Access: Installation for our toolchain.	MySQL: Manual Installation HyperV: Manual			MySQL: Manual Installation HyperV: Manual	MySQL: Manual Installation HyperV: Manual
8 Training & Education	Microsoft Word + Powerpoint: Manual Tortoise svn: Manual Acrobat Reader: Manual		Camtasia Studio 8: Manual	Microsoft Word + Powerpoint: Manual Acrobat Reader: Manual Tortoise svn: Manual		Microsoft Word + Powerpoint: Manual Google Drive: Manual

5.4. Gap Analysis

5.4.1. Level of automation of the toolchain & Information passed

In general, there are three tools providers: IBM, The Reuse Company and Altium. The interoperability and integration between the products of IBM and the Reuse company is ensured but not with Altium.
IBM and the Reuse Company provide standardized ways of accessing (files and services) and consuming the information. However, the interpretation of standards (such as SysML or ReqIf) may differ from tool and another and, in most cases, the tools manage more relevant information that is not exposed and it is critical for processes such as traceability or quality management. This situation also implies that the reusability factor of system artifacts is

decreased. It is possible to first define the interoperability characteristics of the tools to then make a brief evaluation of the integrations needed.
Altium does not provide any standardized way of accessing and consuming information and that is why an adapter will have to be implemented.
According to the table below, each of the tools supporting the engineering process manages a type of system artifact that is available through different data and communication models.

5.4.2. Level of integration with the Arrowhead Framework

Still not connected, in this second year we plan to connect at least Doors, Altium and KnowledgeManager to the Arrowhead Framework.

5.4.3. Missing Tools within the UC

Here, it is important to emphasize the need for connection and access to:

- Requirements (IBM Doors)
- Logical models (IBM Rhapsody)
- Hardware models (Altium Designer)
- Traceability and reuse capabilities (KnowledgeManager)
- Development tool (Eclipse)
- Development tool (Visual Studio)

5.4.4. Missing Tools Outside the UC

N/A

5.5. Declared Tools

The use case will develop OSLC KM-based connectors for three different tools: IBM Doors, IBM Rhapsody and Altium Designer. Afterward, operations for reuse and traceability will be accessible from the KnowledgeManager tool. Moreover, the quality will be a key process to ensure the two previous operations.
The 6 tools specified are the ones specified in D4.1 here renamed. The old ones were IBM Doors, Architecture Definition, Design definition, Implementation, Verification & Validation, Information Management.

Name	This is a tool in the following phase	Compatible with AHF	Inputs	Outputs
IBM Doors	Operation & Commissioning	Not at the moment, we are going to make Arrowhead Framework compliant the second year.	Requirements Altium components, Requirement-Altium components Links	Requirements Requirement-Altium components Links
Requirements Authoring tool	Requirements	Not at the moment, we are going to make Arrowhead Framework compliant the second year.	Requirement specifications, Altium components	Improved requirements specification, Quality reports

IBM Rhapsody	Requirements	Not at the moment, we are going to make Arrowhead Framework compliant the second year.	Requirement specifications	Doors module
Altium designer	Procurement & Engineering	Not at the moment, we are going to make Arrowhead Framework compliant the second year.	Requirements Altium components, Requirement-Altium components Links	Altium components, Requirement-Altium components Links
Verification Studio	Requirements, Functional design	Not at the moment, we are going to make Arrowhead Framework compliant the second year.	Requirements Altium components, Requirement-Altium components Links	Verification and validation reports, Verification and validation metrics
KnowledgeManager	Requirements, Functional design	Not at the moment, we are going to make Arrowhead Framework compliant the second year.	Altium components, Textual patterns, Domain vocabulary, BOM	Ontology with patterns and vocabulary

6. UC-04.1 [ASML, ICTG, TUE]: Interoperability between (modeling) tools for cost effective lithography process integration - CIDT Toolchain

Contact: Ramon Schiffelers [ramon.schiffelers@asml.com], Sven Weber [sven.weber@asml.com], Koen van Wijk [koen.van.wijk@ict.nl], Oscar Reynhout [oscar.reynhout@ict.nl]
The description of each Use Case can be found in D1.2
Here, the toolchains are too complex to be described in a single section, therefore we split it into two. This is the CIDT Engineering process.

6.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 1

AHT-EPP	CIDT Toolchain
1 Requirements	Matlab is used as a scratch book
2 Functional Design	These unprecise specifications are manually translated to a software specification during the procurement and engineering phase
3 Procurement & Engineering	Manually Implemented
4 Deployment & Commissioning	use Linux patching to deploy to the field.
5 Operation & Management	the scenarios are captured in field procedures in Word
6 Maintenance Decommissioning & Recycling	on the software the scenario is retrieved from ClearCase and updated
7 Evolution	
8 Training & Education	field procedures

6.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 1
Phases 5-8 not present

AHT-EPP	CIDT Toolchain
1 Requirements	Matlab: high level and detailed requirement specifications.
2 Functional Design	Matlab: Generated code and interfaces
3 Procurement & Engineering	Detailed component behavior
4 Deployment & Commissioning	

6.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 1
Phases 5-8 not present

AHT-EPP	CIDT Toolchain
1 Requirements	
2 Functional Design	
3 Procurement & Engineering	Matlab -> Python no automation no automation for Word/Visio no automation for Python
4 Deployment & Commissioning	Matlab -> ClearCase Version management

6.4. Gap Analysis

6.4.1. Level of automation of the toolchain & Information passed

The level of automation is low. The phases are linked by unique component names, error codes, document trees per component.

- [Requirements ↔ Functional design: specification documents managed by reviews and approvals]
- [Functional design ↔ Engineering: design document managed by reviews and approvals]
- [Functional design ↔ Engineering: implemented software managed by reviews and approvals and testing]
- [Functional design ↔ Deployment and commissioning, Field Commissioning document managed by reviews and approvals]
- [Engineering ↔ Operation and Management Field operation (FCO) document managed by reviews and approvals]
- [Operation and Management ↔ maintenance: Proprietary ticketing system (AIR), issues managed by priority setting and reviews]
- [Operation and Management ↔ evolution: (AIR) Issues managed by priority setting and reviews]

6.4.2. Level of integration with the Arrowhead Framework

The toolchain is not integrated into the framework yet. We are currently working on that as the next immediate step.

6.4.3. Missing Tools within the UC

In the current, document based workflow, (Word/Visio) specifications are imprecise, ambiguous and incomplete. In the envisioned workflow and supporting toolchain, Stateflow and/or LSAT will be used for unambiguous specification of scenarios, requirements and plant capabilities. The manual translation of functional scenarios into software will be replaced by code generators such as Matlab Coder that transform scenarios in terms of Stateflow specifications into source code. Tools for code generation: Matlab Coder and Simulink Coder. Tools for toolchain integration and data exchange: Jenkins.

6.4.4. Missing Tools Outside the UC

Tools for scenario, requirements and platform specification: Stateflow, Simulink, MATLAB.

6.5. Declared Tools

Stateflow/Simulink/MATLAB enable communication for engineers / stakeholders at the same place, about the same thing, that is executable. Stateflow/Simulink/MATLAB can generate

code. Vcraft is used to generate a skeleton of the integration code, including makefile, including template code. Jenkins is used to make sure that all (integration and quality) checks are ran on every check in. Swish makes sure that the model can interface with existing software components by generating language bindings between Simulink and the existing source code using an existing interface language. The hybrid server makes sure that the model can interface with a live machine.

Name	**This is a tool in the following phase**	**Compatible with AHF**	**Inputs**	**Outputs**
Stateflow/Simulink/MATLAB	1 Requirements, 2 Functional Design, 3 Procurement & Engineering	no	Business Idea	Generated documents, generated code
Jenkins	3 Procurement & Engineering	no	Generated code	Integrated Code
Swish	3 Procurement & Engineering	no	Existing interfaces description	Interface blocks
Vcraft	3 Procurement & Engineering	no	component properties, templating	generated Python code
hybrid server	3 Procurement & Engineering	no	Model	Live connection to machine

7. UC-04.2 [ASML, ICTG, TUE]: Interoperability between (modeling) tools for cost effective lithography process integration - SSVP Toolchain

Contact: Ramon Schiffelers [ramon.schiffelers@asml.com], Sven Weber [sven.weber@asml.com], Koen van Wijk [koen.van.wijk@ict.nl], Oscar Reynhout [oscar.reynhout@ict.nl], Sander Thuijsman [s.b.thuijsman@tue.nl], Ferry Timmers [f.timmers@tue.nl], Alireza Mohammadkhani [a.mohammadkhani@tue.nl], Jeroen Voeten [j.p.m.voeten@tue.nl], Marc Geilen [m.c.w.geilen@tue.nl], Jan Friso Groote [j.f.groote@tue.nl], Michel Reniers [m.a.reniers@tue.nl], Loek Cleophas [l.g.w.a.cleophas@tue.nl]

The description of each Use Case can be found in D1.2
Here, the toolchains are too complex to be described in a single section, therefore we split it into two. This is the SSWP Engineering process.

7.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 6
Only phase 2 is taken into account here.
Definition of the stakeholders can be found in D1.2.

AHT-EPP	LSAT Component	SDF3 Component	CIF Component	mCRL2 Component	PLE Component	Model Translation Component
2 Functional Design	Systems implemented by StkH1 are captured in a model by StkH3 and StkH2.	StkH3 and StkH1 can use the model in the LSAT component to analyze the system and provide timing analysis and guarantees.	Systems implemented by StkH1 are captured in a model by StkH2. System requirements specified in Word documents by StkH3 are formalized by StkH2. CIF is used to synthesize a safe supervisory controller for the system	Systems implemented by StkH1 are captured in a model by StkH2. System requirements specified in Word documents by StkH3 are formalized by StkH2, and verified.	Variability information of the product line is captured in a model by StkH1, StkH2 and StkH3.	

7.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 6
Only phase 2 is taken into account here.

AHT-EPP	LSAT Component	SDF3 Component	CIF Component	mCRL2 Component	PLE Component	Model Translation Component
2 Functional Design	Microsoft Excel + Word: Word documents and informal descriptions of the system LSAT: Moleling	LSAT: Modeling SDF3: Timing Analysis	Microsoft Excel + Word: functional requirements LSAT: modeling CIF: supervisory controller synthesis	Microsoft Excel + Word: functional requirements LSAT: modeling mCRL2: formal verification	LSAT:_x000D_ modeling_x000D_ _x000D_ PLE:_x000D_ State of art PLE techniques	Microsoft Excel + Word: Word documents and informal descriptions of the system LSAT: Modeling

7.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 6
Only phase 2 is taken into account here.

AHT-EPP	LSAT Component	SDF3 Component	CIF Component	mCRL2 Component	PLE Component	Model Translation Component
2 Functional Design	LSAT -> SDF3 SDF3 is used as a plugin in LSAT for analysis of system behavior LSAT -> CIF A translation method has been constructed, not yet implemented	LSAT -> SDF3 SDF3 is used as a plugin in LSAT for analysis of system behavior	CIF -> mcrl2 An integrated CIF to mcrl2 model-to-model transformation is available. CIF -> LSAT CIF models can be inputted in LSAT through a plugin LSAT -> CIF A translation method has been constructed, not yet implemented	Microsoft Excel + Word: manual translation LSAT: manual translation mCRL2: manual translation and operation	LSAT: Manual variability management PLE: Manual variability management	

7.4. Gap Analysis

7.4.1. Level of automation of the toolchain & Information passed

The current level of integration of the toolchain is manual. Some model-to-model transformations are available, either implemented or not. Even for implementation transformations, manual steps are required to communicate between the tools.
Information is contained in models that consist of the relevant information for the respective tools. These models can (in some cases) be converted between the tools, and thus convey the information.

7.4.2. Level of integration with the Arrowhead Framework

The arrowhead framework is not yet used for translating and communicating the models between the tools. Conversion methods have been and are being established. A model translation component can be used in the framework to realize this.

7.4.3. Missing Tools within the UC

In the current, document based workflow, (Word/Visio) specifications are imprecise, ambiguous and incomplete.
In the envisioned workflow and supporting toolchain, LSAT will be used for unambiguous specification of scenarios, requirements and plant capabilities.
Instead of manual design, CIF/SDF3 will be evaluated to synthesize safe and optimal supervisory controllers to enable the execution of multiple scenarios and run-time composition of scenarios on the platform.
Automatic verification and validation procedure using the mCRL2 tools will be developed to ensure and improve quality of designs.
Currently the variability management is not supported in a defined way. Variability management will be supported with PLE techniques to improve reuse of the models.

Tools for scenario, requirements and platform specification: LSAT.
Tools for scenario synthesis: CIF/SDF3
Tools for formal analysis: mCRL2
Tools for product line variability management: PLE Tool

7.4.4. Missing Tools Outside the UC

N/A

7.5. Declared Tools

Recalling D4.1:

LSAT
LSAT (developed by ASML, TNO-ESI and TUE) provides a formal modeling approach for compositional specification of both functionality and timing of manufacturing systems. The performance of the controller can be analyzed and optimized by taking into account the timing characteristics. Since formal semantics are given in terms of a (max, +) state space, various existing performance analysis techniques can be applied.
CIF
CIF (Compositional Interchange Format for hybrid systems, developed by TUE) is an automata-based modeling language for the specification of a discrete event, timed, and hybrid systems. The CIF tooling supports the entire development process of controllers, including, among others specification, supervisory controller synthesis, simulation-based validation and visualization, verification, real-time testing, and code generation.
SDF3

SDF3 (developed by TUE) is a tool for the analysis and synthesis of Synchronous DataFlow Graphs (SDFGs). It includes an extensive library of SDFG analysis and transformation algorithms as well as functionality to visualize them. It also includes analysis algorithms for switching (max,+) models. The tool can also create SDFG benchmarks that mimic DSP or multimedia applications.
mCRL2
mCRL2 (developed by TUE in collaboration with the University of Twente) is a formal specification language with an associated toolset. The toolset can be used for modeling, validation and verification of concurrent systems and protocols. The toolset supports a collection of tools for linearisation, simulation, state-space exploration and generation, and tools to optimize and analyze specifications. Moreover, state spaces can be manipulated, visualized and analyzed.
PLE tool:
The envisioned PLETool is a tool for product line variability management. PLETool provides the infrastructure for managing the variability in a product line. The tool incorporates different components to support variability modeling, analysis and configuration of the variability model, and mechanisms for product derivation. It will be aligned with the various modeling languages employed with other tools in the use case, to ensure proper variability support.

Model translation tool:
The model translation tools provide a centralized component for translating domain specific models, used by the other tools part of the framework, back and forth, allowing for multi-disciplinary interoperability between the different tooling.

Name	This is a tool in the following phase	Compatible with AHF	Inputs	Outputs
LSAT Component	Functional Design	Currently not, but integration is currently being evaluated.	Word documents and informal descriptions of the system to be modeled	LSAT model
SDF3 Component	Functional Design	Currently not, but integration is currently being evaluated.	LSAT Model	Timing analysis and timing guarantees
CIF Component	Functional Design	Currently not, but integration is currently being evaluated.	LSAT - system behavioral model Informal / formal requirements	Safe supervisory controller for the input system
mCRL2 Component	Functional Design	Currently not, but integration is currently being evaluated.	mCRL2 - model specification language LTS / FSM / AUT - state space specification MCF - formalized requirements	LTS / FSM / AUT - generated state spaces Diagnosis - result of verification and analysis

PLE Component	Functional Design	Currently not, will be compatible with AHF.	Variability model of the product line, LSAT model product family	Derived LSAT model product
Model Translation Component	Functional Design	Currently not, will be compatible with AHF.	Models from the other tools in the usecase	Models for the other tools in the usecase

8. UC-05 [KAI, IFAT, IFAG, UC3M, REUSE]: Support quick and reliable decision making in the semiconductor industry

Contact: Anja Zernig, Patrick Moder [anja.zernig@k-ai.at, patrick.moder@infineon.com]
The description of each Use Case can be found in D1.2

8.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 4

AHT-EPP	UC05-specific ontology	DR	TePEx	WHF
1 Requirements	Define requirements based on documents and expert interviews	-	Requirements exist for the "stand-alone version". These requirements will be extended based on user feedback	Requirements exist for the "stand-alone version". These requirements will be extended based on user feedback
2 Functional Design	DrOWLings and basic modeling tools are used to gain a general understanding of the structure	only situative (for merging preparation)	implicitly considered in "3 Procurement & Engineering"	implicitly considered in "3 Procurement & Engineering"
3 Procurement & Engineering	Model development and instantiation	-	Generalization of TePEx with and hence, for a broad product palette	Generalization of WHF with and hence, for a broad product palette
4 Deployment & Commissioning	Integration into DR	Integration into DR	Integration of TePEx into existing IT framework (existing toolchain)	Integration of WHF into existing IT framework (existing toolchain)
5 Operation & Management	WebVOWL for visualization	WebVOWL for visualization	implicitly considered in "3 and 4 "	implicitly considered in "3 and 4 "
6 Maintenance Decommissioning & Recycling	WebVOWL for maintenance	WebVOWL for maintenance	implicitly considered in "3 and 4 "	implicitly considered in "3 and 4 "
7 Evolution	WebVOWL for evolvement	WebVOWL for evolvement	implicitly considered in "3 and 4 "	implicitly considered in "3 and 4 "
8 Training & Education	WebVOWL for training	WebVOWL for training	TePEx is documented in a Master Thesis. A manual for the end-user will be written in the course of this project	WHF is documented in a PhD thesis. A manual for the end-user will be written in the course of this project

8.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 4

AHT-EPP	UC05-specific ontology	DR	TePEx	WHF
1 Requirements	excel	-	Microsoft Excel + Word: Document with the requirements elicitation.	Microsoft Excel + Word: Document with the requirements elicitation.
2 Functional Design	excel; yED	-	R, RStudio, RShiny: programming language for algorithm Git/BitBucket: versioning system	R, RStudio, RShiny: programming language for algorithm Git/BitBucket: versioning system
3 Procurement & Engineering	Protégé; Apache Jena Fuseki	-	R, RStudio, RShiny: programming language for algorithm Git/BitBucket: versioning system	R, RStudio, RShiny: programming language for algorithm Git/BitBucket: versioning system
4 Deployment & Commissioning	Protégé; Apache Jena Fuseki	Protégé; Apache Jena Fuseki	using internal/inhouse company software and infrastructure	using internal/inhouse company software and infrastructure
5 Operation & Management	Protégé; WebVOWL	Protégé; WebVOWL	using internal/inhouse company software and infrastructure	using internal/inhouse company software and infrastructure
6 Maintenance Decommissioning & Recycling	Protégé; WebVOWL	Protégé; WebVOWL	using internal/inhouse company software and infrastructure	using internal/inhouse company software and infrastructure
7 Evolution	Protégé; WebVOWL	Protégé; WebVOWL	using internal/inhouse company software and infrastructure	using internal/inhouse company software and infrastructure
8 Training & Education	excel; WebVOWL	excel; WebVOWL	Acrobat Reader: writing the manual	Acrobat Reader: writing the manual

8.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 4

AHT-EPP	UC05-specific ontology	DR	TePEx	WHF

1 Requirements	highly manual	-	Microsoft Excel + Word: manual	Microsoft Excel + Word: manual
2 Functional Design	highly manual	-	R, RStudio, RShiny: mainly automated Git/BitBucket: mainly automated	R, RStudio, RShiny: mainly automated Git/BitBucket: mainly automated
3 Procurement & Engineering	manual, barely automated input from 1 or 2	-	R, RStudio, RShiny: mainly automated Git/BitBucket: mainly automated	R, RStudio, RShiny: mainly automated Git/BitBucket: mainly automated
4 Deployment & Commissioning	mainly automated	mainly automated	using internal/inhouse company software and infrastructure: mainly automated	using internal/inhouse company software and infrastructure: mainly automated
5 Operation & Management	mainly automated (visualization, inference)	mainly automated (visualization, inference)	using internal/inhouse company software and infrastructure: mainly automated	using internal/inhouse company software and infrastructure: mainly automated
6 Maintenance Decommissioning & Recycling	depends on maintenance tasks (automated capabilities in place but not extensively)	depends on maintenance tasks (automated capabilities in place but not extensively)	using internal/inhouse company software and infrastructure: mainly automated	using internal/inhouse company software and infrastructure: mainly automated
7 Evolution	rather manual (feedback loops, re-modeling)	rather manual (feedback loops, re-modeling)	using internal/inhouse company software and infrastructure: mainly automated	using internal/inhouse company software and infrastructure: mainly automated
8 Training & Education	depends on user expertise and intention (automated capabilities in place for visualization preferences and re-adjustments)	depends on user expertise and intention (automated capabilities in place for visualization preferences and re-adjustments)	Acrobat Reader: manual	Acrobat Reader: manual

8.4. Gap Analysis

8.4.1. Level of automation of the toolchain & Information passed

Currently, TePEx and WHF act as stand-alone tools for post-processing analysis, without automated interoperability to related data sources and real-time feedback. To achieve this, the integration of TePEx and WHF into appropriate existing platforms (existing toolchains) is

necessary and planned in the course of this project. Therefore, it is important to know the requirements of the system (Requirements), to guarantee a valid methodology of the algorithms applicable to a broad product palette (Engineering and Procurement) and the hand-over to the existing infrastructure (Deployment). Of course, also other EPPs like Maintenance and Training are considered, but the main focus is on Requirements, Engineering & Procurement and Deployment.

For DR:
Requirements and functional design: the knowledge gathering process and the initial concept structuring process require high manual effort. However, existing ontologies (such as DR, e.g.) can be exploited to achieve knowledge build-up faster. From our point of view, it is rather unlikely to come to a more automated solution here without major shortcomings w.r.t. data quality and validity. The same holds for the Engineering Phase. Some approaches are currently investigated within the team, how a rather automated model design can be achieved. However, especially when it comes to specific domain descriptions, these generalized approaches are barely applicable.
Deployment, Operation and Management are mainly automated. This is the major advantage of a semantic concept description in our opinion.
Maintenance, evolution and training are on the one hand manual, with regard to feedback loops, capturing user experience, providing training material. On the other hand, automated tools exist that allow fast re-modeling, direct feedback, and customized visualization. Room for improvement in terms of automation is seen for further customization.
For the global aim of integrating TePEx and WHF, EPPs have to be passed multiple time, since it's a cycle: Requirements (on TePEx and WHF) --> Procurement & Engineering (improvement of TePEx and WHF) --> Deployment and Commissioning (integration of TePEx and WHF) --> Requirements (again for the next dataset/product under investigation). As soon as TePEx and WHF are integrated, the Training phase comes into play, by writing the final manual.]

For DR:
1-->2-->3: unstructured or csv
3-->4-->5: owl or rdf
5-->6-->7-->8: owl, rdf, json
6/7/8-->3: unstructured, csv, owl, rdf

8.4.2. Level of integration with the Arrowhead Framework

No points of contact with the Arrowhead Framework exist at the beginning of this project.

Currently no actual integration, however, dedicated interfaces in place. And the Arrowhead Framework is conceptually modeled as a semantic description (as part of DR).

8.4.3. Missing Tools within the UC

Tools for data analysis and the respective underlying algorithms (for automating part of the rule-wise description of class relations.

TePEx, WHF and DR (including the UC-05 specific ontology) are the core components of UC05 within the Arrowhead Tools project, and hence, indispensable. Tool selection is based on best-practices and experience, so might be exchangeable (however not intended).

8.4.4. Missing Tools Outside the UC

Nothing too urgent. However, if someone has tools for automatic model development or customized training/feedback automation in place, this might be of interest.

8.5. Declared Tools

UC05-specific ontology will be developed and later matched (integrated) with DR.

Name	This is a tool in the following phase	Compatible with AHF	Inputs	Outputs
UC05-specific ontology	1--8	yes	data (maybe unstructured)	harmonized data (owl/rdf/json)
DR	2; 4--8	yes	data (maybe unstructured)	harmonized data (owl/rdf/json)
TePEx	3, 4	in principle yes, AHF will be evaluated and TePEx integrated	wafer test data	TePEx output visualization
WHF	3, 4	in principle yes, but we see no need for integration	wafer test data	WHF output visualization

9. UC-06 [LBB, LQT, POD, ADV, LTU]: Production preparation toolchain integration

Contact: Lars Oskarsson [lars.oskarsson@lindbacks.se]
The description of each Use Case can be found in D1.2

9.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: N/A

AHT-EPP	**NOT PROVIDED**

9.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: N/A

AHT-EPP	**NOT PROVIDED**

9.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: N/A

AHT-EPP	**NOT PROVIDED**

9.4. Gap Analysis

9.4.1. Level of automation of the toolchain & Information passed

Currently, all transfer of information between tools are done manually. With the change in CAD software (from DDS to Vertex BD), functional design is now automated as the change is from 2D to 3D. This becomes an integrated or internal information transfer.

Currently, the type of information passed between the tools are architectural drawings, either in the form of PDF documents or in IFC-BIM format.

9.4.2. Level of integration with the Arrowhead Framework

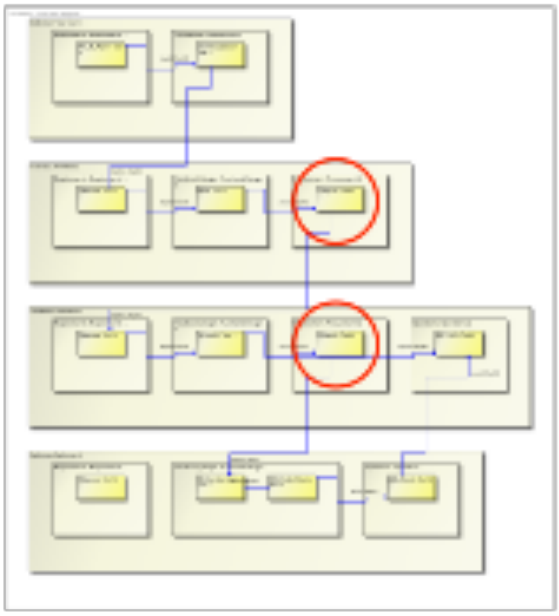
Currently the Arrowhead Framework is not yet being used. Lindbäcks has a local cloud up and running and PodComp is about to set up their own. What is missing are the systems that

monitor the new output of the tools and look for the service provider that would process these outputs both within the stakeholders and across stakeholders.

9.4.3. Missing Tools within the UC

PodComp is developing a tool that takes 3D drawings and generates the machine files for ABB robots to cut the walls and floors for the bathroom pods. The machine files include ventilation, electricity and plumbing holes.

Lundqvist is working with its 3D Configurator to provide files to PodComp's new robot machine file generator. Lindbäcks has already made the move to Vertex BD and can export the necessary 3D files for PodComp.



Multi stakeholders' connected engineering phases and tools.

The tools that are indispensable for a SOA tool chain include the service provider that registers its services with the Service Registry. Similarly, the tool which queries and consumes services needs to be developed to form that tool chain (see the aside figure with red circles).

9.4.4. Missing Tools Outside the UC

The tool that needs to be developed is the one that checks the generation of new files to be processed by the next tool. This is currently being done manually and should be done using service-oriented architecture. If we consider long term support, BNearIT should be the one who develops this tool, but they are not officially part of the use case. Nonetheless, they have been supporting it.

9.5. Declared Tools

Declared tools have not been provided for this iteration. The information from D4.1 are used to fill the following table. Tools are:

Vertex - drawing tool with BIM-abilities is being implemented at Lindbäcks to replace DDS
3D configurator - tool for making machine files understandable by ABB robots from ifc files

Name	This is a tool in the following phase	Compatible with AHF	Inputs	Outputs

Vertex	Requirements, Fucntional Design	No	dwg drawing (EP-I1)	ifc-files (EP-O2)
3D Configu rator	Operation & Management	Yes	ifc-file (EP-I5)	machine file (EP-O5)

10. UC-07 [FAUT, IKERLAN]: CNC machine automation

Contact: Carlos Yurre [yurre@aotek.es]
The description of each Use Case can be found in D1.2

10.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 2

MT : Machine Tools
MTBld : MT Builder
MTUser: MT Operator-programmer

AHT-EPP	Machine Tool Construction	Part Production
1 Requirements	The MTBld decides to construct a new model of MT or customization of an existing model for a customer. These reqs. usually determine dimensions, power, speeds and type of CNC.	The MtUser receives an order to produce a batch of parts. He can receive the requirements in many ways, from a blueprint and material to 3d geometry in a (hopefully) standard format.
2 Functional Design	MTBld designs the machine mechanically. CAD systems and FEM applications are used. To select components from the CNC supplier, Cad drawings and Datasheets are available. In some cases, the structural analysis provides resonance frequencies from the design phase.	The programmer decides the best way of producing the part in the machine. He can import dxf files (or any other standard format), select geometric elements are of interest, define the operations to be done with them (drilling, machining, threading…), modify the geometries or make new ones from a blueprint.
3 Procurement & Engineering	MTBld places orders for all the components. Depending on the company, many of the components are bought or subcontracted to their parties. CNC, drives, motors and linear scale models are somewhat complex and choice is usually done assessed by commercial people with technical background. After the selection is made, the order is placed and the material is served. For new models or new customers, technical advice from Fagor Automation is also offered for configuring and tuning. The engineering team will develop the documentation of the HW and SW developed for the ECU that will be provided as a reference manual to the StkH 1. A working prototype of the ECU will be tested in a real SBS machine for the quality test in the vendor laboratories.	The programmer can iteratively refine the program using different features and machining conditions using the geometric aids and the technology tables. The phase produces a G-Code file (or several files) and the needed tools. These are retrieved and must be loaded in the tool magazine.
4 Deployment & Commissioning	MTBld, after the MT is built, needs to diagnose for proper work of the system, align axes, program the PLC, connect the Ios to the fieldbuses, configure them… After that, the tuning of the control loops is carried out.	The operator downloads the program (frequently the program is written or modified at the machine) , does the setup: putting the raw material, the tools, setting the offsets (zeroing the part geometry), all that is needed to start operation.

	In this phase, for new models, it is common that both the MTBld technician and the Fagor Automation service people work together. For the next machines, only the tuning optimization must be done, configuration is copied from the first machine.	
5 Operation & Management	This phase includes the first 5 Eps of the part piece production	The operator can start production for the number of pieces of the batch. For the relevant use case, the operator must care for final adjustment of the feeds and speeds, adapting to material and tool wear. Usually, every new part requires a new setup, but zeroing and measuring can be automated for machines that integrate touch probes.
6 Maintenance Decommissioning & Recycling	This task can be done by different actors, and one of them is the MTBld. There can be a maintenance contract. Data from the commissioning phase is used as a baseline to diagnose the machine. Some of the tuning tools are relevant here. A comparison with baseline gives a "health" assessment.	The operator usually "monitors" the health of the machine. This is currently substituted (or at least supplemented) by logging systems and automatic diagnosis systems (with learning algorithms, etc.) Commissioning tools are not directly used by the operator but specific tests based on the same tools can be carried out automatically under operator supervision.
7 Evolution	New CNC versions would incorporate new features of interest to the MTBld and also correct software errors. New machines should use the last versions. Moreover, on occasion of maintenance work, or when any error is important enough for a customer, the CNC software version must be upgraded. This is a delicate operation that is done manually (giving access to directories, taking into account of options...)	The MTUser can report field errors and, when those are solved, can download new versions. Moreover, the MTUser can decide to download, under license, new applications, tools or canned cycles of interest for his work. This requires an application designed specifically for that matter.
8 Training & Education	The MTBld can use the CNC Simulator incorporating the kinematics and drawings of its own machine. This produces, in fact, a digital twin useful for the programming of the machine and, much important, to detect collision between machine and part, etc…	The simulator is in this case very helpful. Many licenses are today sold for training and education in CNC programming. Regarding this, the basic version is freely downloadable and is used at professional schools. The look and feel is just as that of the CNC, what improves familiarity for students.

10.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 2

AHT-EPP	Machine Tool Construction	Part Production
1 Requirements		dxf, blueprint.

		Pdf or .doc with materials, number of parts, etc…
2 Functional Design		CNC Tools: (our tools) Global and local parameter tables editor Technology tables editor CNC program editor Profile editor and viewer Interactive programming dxf importer…
3 Procurement & Engineering		CNC Tools(our tools): Tool table Tool Magazine CNC Simulator Technology tables editor Canned Cycles
4 Deployment & Commissioning	CNC tools: (our tools) Parameter Editor PLC programming and compiling. Logic analayzer Oscilloscope Fine Tune Circularity adjustment	CNC Tools: (our tools) CNC operation modes (manual, MDI, zeroing, automatic…) Offset table editor Kinematic table adjustment and editor MTBld defined tools: status of periphery...
5 Operation & Management		CNC tools: (our tools) CNC operation modes (automatic) Program viewer graphical view Canned cycles and routines made by the MTBld in G-Code Operating Modes defined by the MTBld.
6 Maintenance Decommissioning & Recycling	CNC tools: (our tools) Tuning tools: Oscilloscope, Diagnosis MTBld procedures dedicated and running on the CNC Specific MTBld Tools (Python, Matlab…)	
7 Evolution	Reports sent via email. Acrobat Reader: New features of CNC, corrected errors.	Acrobat Reader: analysis of PDF reports generated by CNC manufacturer and MTBld.

8 Training & Education		CNC Simulator: Basically a complete CNC without the control hardware

10.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 2

AHT-EPP	**Machine Tool Construction**	**Part Production**
1 Requirements		dxf, manual trigger, automatic connection of data from importer to G-code blueprint. Manual interpretation Acrobat or word or libreoffice: Manual
2 Functional Design		CNC Tools: (our tools) Manual trigger of actions. Some actions chained (editing a program selects it for running) Same for other stored information. Command to automatically store and compress all the configuration data of a machine to clone it . (or for debugging at simulators) Files with G-code must be manually moved or copied from CNCs or office PCs.
3 Procurement & Engineering		CNC Tools(our tools): Same as phase 2
4 Deployment & Commissioning	CNC Tools: Integration of tools is default inside the CNC application. Actions are triggered manually, internal tools communicate via proprietary formats or direct memory sharing. Oscilloscope data is not integrated in fin tune application. Fine-tune is an external tool and can read Machine tool data also in proprietary format.	CNC Tools: (our tools) manual triggering, integrated handling of data. Tables don't need to be saved usually. MTBld defined tools: manual
5 Operation & Management		CNC tools: (our tools) Some automatic (integrated) behavior between tools. Editing a table has immediate effect under some circumstances.

6 Maintenance Decommissioning & Recycling	Acrobat Reader: manuals Oscilloscope: manually import stored data from commissioning to compare Fine Tune:Manual import of stored data	
7 Evolution	Acrobat Reader: manual	Acrobat Reader: Manual
8 Training & Education		CNC Simulator: Operation is of course manual. Machine Data can be imported manually from real machines for configuration, etc…

10.4. Gap Analysis

10.4.1. Level of automation of the toolchain & Information passed

The deployment, programming and operation phases in this use case can be done (and are usually done) by different people. Inside deployment, the tools are well integrated (being the tuning tool an exception) but they lack standardization in interfaces and in input/output data. The simulation tool is very well suited for CNC programming but lacks integration with the engineering phases of deployment and commissioning.

Configuring the HMI is today done with a different tool but results can be used immediately. It is built on outdated technologies and can't be integrated with the new interfaces. The automation of interactions is rather manual, that is copying of files, translations, templates for programming, etc., while the tools use the same platform. There's no "piece part" concept. The current CNC is oriented to program editing and the inclusion of technologies, machining conditions, etc. is ad-hoc.

The information of new tools is mainly passed by files, being many of them .xml files, whilst there is no XSD or DTD for them. Legacy files and parameter descriptions can be found in other formats, from plain text or .csv files to binary formats. In some cases, the components interact through internal data and shared memory. Persistence can also be compressed. Access to some data is done via password, which depends on the access level.

10.4.2. Level of integration with the Arrowhead Framework

Tool	Integration
1.1 Topology validator (B1)	Y
1.2 Topology Editor (B1)	N

1.3 Machine Parameter validator (B3)	Y
1.4 Legacy Parameter Converter (B3)	Y
1.5 Connection Editor (B2)	N
1.6 Parameter Editor (B3)	N
2.1 Legacy scope data conversion (B5)	Y
2.2 Fast data acquisition Tool (B5,B8)	N
2.3 Data acquisition configuration tool (B5,B8)	N
2.4 Data plotting tool (B5,B8)	N
2.5 Transfer Function Identification (B5)	Y
2.6 Control Loop Optimizer Tool (B5)	Y
3.1 dxf importer to geometry (B7)	Y
3.2 Legacy profile and ISO (G-code) importer (B7,B10)	Y
3.3 Technology tables reader and validator (B6,B7)	Y
3.4 ISO(G-code) generator from geometry (B6,B7)	Y
3.5 Part description reader and validator (B6,B7)	Y
3.6 Machining strategies reader and validator (B7)	Y
3.7 Geometry Editor (B6,B10)	N
3.8 Advanced CNC operations generator (B6,B10)	N

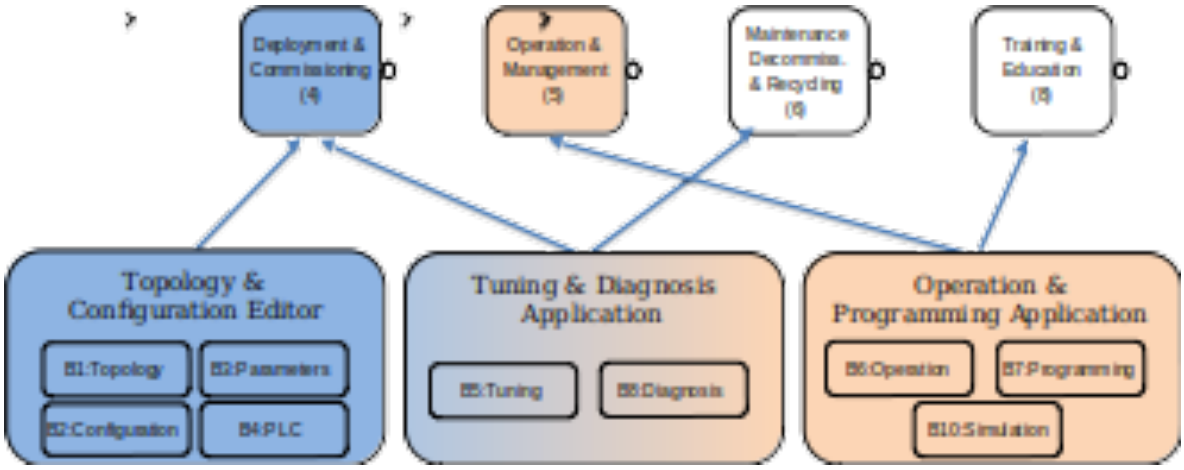
10.4.3. Missing Tools within the UC

While having strong points, tight integration as found in the current CNC has some drawbacks: it is a barrier for third-party companies that could add value to our customers' machines by

integrating their algorithms. This same third party could add a small fee paid to FAUT. This is a business model already used in some CNC manufacturers.

The tight integration is a bottleneck today as it is not suitable for cloud-based tools or information handling, connectivity, etc… Proprietary file formats and hidden data exchange protect know-how and are optimum in size and speed…but are not suitable for use of open-source software and/or standard tools. This is a big issue.

Modularity is missing or inadequate today, due mainly to the way that the applications have followed to evolve. Even when code is reused, intermediate data is not persisted and is done frequently in binary format and in memory. This makes it difficult to inherit behaviors and methods from legacy tools or subroutines. In some of the applications, rules for data transformation are “hardcoded” in the programs, making it difficult to change behavior or optimize without complete reload of the program. These issues will also be addressed during the project.



A decomposition of the applications in smaller activities and tools follows (as shown in the figure).

The decomposition of the tools associated with the identified blocks has been aggregated in three future toolchains (possibly implemented as applications) considering the users. The first toolchain addresses blocks B1 to B4, while B4 will not be changed from the current behavior. These tools will be used by the MTBld or by Fagor Automation people for machine adaptation. The second toolchain is devoted mainly to loop control tuning and machine diagnosis. This tool will be used by the last mentioned actors, but chances are that some MTUsers or even third party companies could use it too. The third toolchain is devoted to MTUser, the programmer and/or operator. It is a very ambitious tool and a big improvement over the current program edition. The upgrade/update block hasn’t been included as the requirements are unclear, but it could be incorporated during the project. Inputs and outputs of the tools are specified in the table below, but we have decomposed the above tools, to gain modularity and project control in smaller tools, following the tools definition. Further analysis during development could discover other small tools that can be integrated in one (or more) of the three toolchains or the output of these small tools could be changed from JSON to XML or any other standard format.

All the described toolchains are necessary, as they address already identified (listed above) tasks. The automatic setup and advanced programming tools seem to have more added value and better return

10.4.4. Missing Tools Outside the UC

We haven't identified those tools today. It seems that 3D geometry readers or geometry web renderers could be good candidates. We are exploring open source projects in that direction. Step-nc or express generic readers could be of interest, but we haven't explored the real need yet.

10.5. Declared Tools

Name	This is a tool in the following phase	Compatible with AHF	Inputs	Outputs
T1.1 Topology validator (B1)	MTBld, Commissioning	Possibly yes	a) stored or detected topology: XML	a) Javascript Object
T1.2 Topology Editor (B1)	MTBld, Commissioning	No	a) Javascript Object b) User Input	a) modified topology: XML
T1.3 Machine Parameter validator (B3)	MTBld, Commissioning	Possibly yes	a) Stored Parameter file (XML) b) Parameter Descriptors (XML)	a) modified Parameter File: XML
T1.4 Legacy Parameter Converter (B3)	MTBld, Commissioning	Possibly yes	a) Legacy Parameter file (TEXT) b) Parameter Descriptors (XML)	a) Parameter File: XML
T1.5 Connection Editor (B2)	MTBld, Commissioning	No	a) System Configuration (Javascript Object) b) Device descriptors (XML)	a) Connection Map(XML) b) Parameter File(XML)
T1.6 Parameter Editor (B3)	MTBld, Commissioning MTUser, Commissioning	No	a) Parameter file (XML) b) Parameter Descriptors (XML)	a) Parameter File: XML
T2.1 Legacy scope data conversion (B5)	MTBld, Commissioning	Possibly yes	a) Acquired Data and Configuration (Matlab.m)	a) Data Acquisition Data file (JSON?) b) Data Acquisition Configuration File (JSON, XML?)

T2.2 Fast data acquisition Tool (B5, B8)	MTBld, commissioning MTBld, maintenance MTUser, maintenance	No	a) Data Acquisition Configuration File (JSON, XML?)	a) Acquired Data (bin) b) Persisted Acquired Data(JSON)
T2.3 Data acquisition configuration tool (B5, B8)	MTBld, commissioning MTBld, maintenance MTUser, maintenance	No	a) Data Mapper File (XML) b) user interaction	a) Data Acquisition Configuration File (JSON, XML?)
T2.4 Data plotting tool (B5, B8)	MTBld, commissioning MTBld, maintenance MTUser, maintenance	No	a) Acquired Data (bin) b) Persisted Acquired Data(JSON) c) Plot configuration (JSON)	a) Plotted Data (HTML) b) Plot configuration (JSON)
T2.5 Transfer Function Identification (B5)	MTBld , Comissioning	Possibly yes	a) Persisted Acquired Data(JSON) b) Configuration data (JSON, XML)	a) Linear Model (JSON, XML)
T2.6 Control Loop Optimizer Tool (B5)	MTBld, Commissioning	Possibly yes	a) Persisted Acquired Data(JSON Linear Model (JSON, XML) b) Strategy(XML)	a) Parameter File(XML)
T3.1 dxf importer to geometry (B7)	MTUser, Part Design MTUser, Part Engineering	Possibly yes, at least partially	a) dxf file (TEXT, 2013)	a) Geometry Description File (JSON)
T3.2 Legacy profile and ISO (G-code) importer to geometry (B7, B10)	MTUser, Part Design MTUser Training & Education	Possibly yes, at least partially	a) ISO (G-code)	a) Geometry Description File (JSON)
T3.3 Technology Tables reader and validator (B6, B7)	MTUser, Part Design MTUser, Part Engineering	Possibly yes	a) Technology Table (XML)	a) Javascript Object
T3.4 ISO(G-code) generator from geometry (B6, B7)	MTUser, Part Design MTUser, Part Engineering	No	a) Geometry Description File (JSON) b) options, style, machine (JSON)	a) ISO(G-code) File(Text)

T3.5 Part description reader and validator (against tools…) (B6, B7)	MTUser, Part Design MTUser, Part Engineering	Possibly yes, at least partially	a) Part piece description (STEP?, XML...)	a) Javascript Object
T3.6 Machining strategies reader and validator (against machine, material, tools) (B7)	MTUser, Part Design MTUser, Part Engineering	No	a) Strategies File (XML, JSON?) b) Geometry Description File (JSON) c) Part Piece description (Javascript Object)	a) ISO(G-code) file(text) b) Advanced Operations File(JSON)
T3.7 Geometry Editor (B6, B10)	MTUser, Part Design MTUser Training & Education	No	a) Geometry Description File (JSON) b) user input	a) Geometry Description File (JSON)
T3.8 Advanced CNC operations generator (B6, B10)	MTUser Training & Education	No	a) System Configuration (Javascript Object) b) Device descriptors (XML)	a) Advanced Operations File (JSON)

11. UC-08.1 [ST-I, Eurotech, IUNET, POLITO, REPLY, BEIA, ROP] SoS engineering of IoT edge devices (Environmental Monitoring)

Contact: Maurizio Griva [m.griva@reply.it]
The description of each Use Case can be found in D1.2

11.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 4

AHT-EPP	Measurement Services	Edge Computing	Vital-IoT	Robofuse
1 Requirements	Collection of User Requirements on Excel Spreadsheet	Collection of user and edge processing requirements together with integration ones to dispatch data to the cloud (e.g. JSON interface format).	Collection of user and application requirements for GUI and interfacing Arrowhead Framework	Collection of user and application requirements for GUI and interfacing Arrowhead Framework
2 Functional Design	Definition of the main points of software development: protocol to use. Production of a specific documentation to drive the development	Definition of the main points of software development: protocol to use. Production of specific documentation to drive the development	Definition of GUI and interfaces among involved systems. Production of specific documentation to drive the development	Definition of GUI and interfaces among involved systems. Production of specific documentation to drive the development
3 Procurement & Engineering	Purchase of unavailable sensors and development of software code for interfaces to send data. Development Test of implemented functions	Purchase of unavailable industrial PC to run EdgeX framework and related pieces of code. Development of the edge processing software code.	Development of code to adapt GUI and internal process of discovery to involve Arrowhead Frameworks architecture services.	Development of code to adapt GUI and internal process of discovery to involve Arrowhead Frameworks architecture services.
4 Deployment & Commissioning	Deployment of software on the different boards and sensors	Deployment of the software code by means of Dockerization	Deployment of the software by means of Dockerization	Deployment of the software by ….
5 Operation & Management	Use of Jira for tracking bugs reported from the operational field and the management phase	Use of Jira for tracking bugs reported from the operational field and the management phase	Use of Jira for tracking bugs reported from the operational field and the management phase	Use of TAIGA for tracking bugs reported from the operational field and the management phase

6 Maintenance Decommissioning & Recycling	Manual tracking of bugs discovered and reported by users Dedicated resolution of reported bugs	Manual tracking of bugs discovered and reported by users Dedicated resolution of reported bugs	End/User usage test to ensure the correct performance of GUI Manual tracking of bugs discovered and reported by users Dedicated resolution of reported bugs	End/User usage test to ensure the correct performance of GUI Manual tracking of bugs discovered and reported by users Dedicated resolution of reported bugs
7 Evolution	Software evolution is driven by: 1) Aggregation of major bugs to solve via a complete software evolution rather than a discrete bug resolution 2) Strategic enhancements of sensor code to introduce new features	Software evolution is driven by: 1) Aggregation of major bugs to solve via a complete software evolution rather than a discrete bug resolution 2) Strategic enhancements of Edge processing code to introduce new features	Software evolution is mainly driven by strategic enhancements of GUI and Back-End code to introduce new features	Software evolution is mainly driven by strategic enhancements of GUI and Back-End code to introduce new features
8 Training & Education	Production of installation manuals and training document for usage and configuration	Production of installation manuals and training document for usage and configuration	Production of installation manuals and training document for usage and configuration	Production of installation manuals and training document for usage and configuration

11.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 4

AHT-EPP	**Measurement Services**	**Edge Computing**	**Vital-IoT**	**Robofuse**
1 Requirements	Excel, Word	Excel, Word	Excel, Word	Excel, Word
2 Functional Design	Excel, Word	Excel, Word	Excel, Word	Excel, Word, UML
3 Procurement & Engineering	Visual Studio Code, Notepad++	Visual Studio Code, Notepad++	Webstorm, Robo3T, Postman	Postman,
4 Deployment & Commissioning	Python, C, C++, Docker, K3S	EdgeX Framework, K3S, C++, Go	Docker, K3S, typescript, python	PHP, typescript, python

5 Operation & Management	Jira	Jira	Jira	Taiga
6 Maintenance Decommissioning & Recycling	Jira, C, C++, Python, Docker, K3S	Jira, C++, Go, EdgeX Framework, K3S	Jira, Webstorm, Robo3T, Postman	Taiga, Postman
7 Evolution	Jira, C, C++, Python, Docker, K3S, Excel, Word, Visual Studio Code, Notepad++	Jira, EdgeX Framework, K3S, C++, Go	Jira, Webstorm, Robo3T, Postman	Taiga, Postman
8 Training & Education	Google Drive, MS Office	Google Drive, MS Office	Google Drive, MS Office	On-line Knowledge base

11.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 4

AHT-EPP	**Measurement Services**	**Edge Computing**	**Vital-IoT**	**Robofuse**
1 Requirements	Management of surveys collected from users are manually curated and analyzed, the process can be partially automated.	Manual Activity, partially automatable	Manual Activity, partially automatable	Manual Activity,
2 Functional Design	Manual activity, generation of the technical specifications from performance requirements.	Semi-automated activity, generation of the technical specifications from performance requirements.	Semi-automated (code design and modeling, tests generation are manual)	Manual Activity,
3 Procurement & Engineering	Manual phase of development	Manual phase of development	Manual phase of development	Manual phase of development
4 Deployment & Commissioning	Semi-automated deployment	Semi-automated deployment	Semi-automated deployment	Semi-automated deployment
5 Operation & Management	Semi-automated phase	Semi-automated phase	Semi-automated phase	Semi-automated phase
6 Maintenance Decommissioning & Recycling	Semi-automated during the tracking, manual in the resolution of bugs	Semi-automated during the tracking, manual in the resolution of bugs	Semi-automated during the tracking, manual in the resolution of bugs	Semi-automated during the tracking, manual in the resolution of bugs

7 Evolution	semi-automated tracking of bugs and activities, manual resolution/development of CRs, semi-automated deployment	semi-automated tracking of bugs and activities, manual resolution/development of CRs, semi-automated deployment	semi-automated tracking of bugs and activities, manual resolution/development of CRs, semi-automated deployment	semi-automated tracking of bugs and activities, manual resolution/development of CRs, semi-automated deployment
8 Training & Education	Manual	Manual	Manual	Manual

11.4. Gap Analysis

11.4.1. Level of automation of the toolchain & Information passed

Measurement services: In the Requirement and specification phase, the collection and analysis of surveys for acquiring useful data are made manually. Then the info is manually included in the procedures collected in the functional design phase. Information is passed using databases and text documents. The type of information is sensor data transmitted through the MQTT protocol. The data is stored in MySQL databases.

Edge Computing: The level of the initial integration of the tools used in this toolchain is very low; a certain level of integration is available in the deployment phase of EdgeX Foundry framework where Dockerization and K3S technologies are used. Information is passed using databases and text documents.

Vital-IoT: The level of the initial integration of the tools used in this toolchain is very low; integration is achieved in the deployment phase by using Docker or K3S technologies. Information is passed using databases and text documents.

Robofuse: The level of the initial integration of the tools used in this toolchain is very low. Information is passed using databases and text documents.

11.4.2. Level of integration with the Arrowhead Framework

At M0 there is no use of the AHF, but during the project we will develop several tools, implementing in the end service providers and service consumers from the perspective of the AHF architecture:

Specifically:

- Temperature, Humidity, PM10 and PM2.5 sensors will result in Service Providers
- Vital-IoT, the legacy tool provided by REPLY, will be an example of Service Consumer
- RoboFuse, tool provided by ROPARDO, will be integrated with Arrow Head Tools, compliant with AHF architecture justifying the “System of Systems” definition of the Use Case

11.4.3. Missing Tools within the UC

At the architecture level, it would be useful to have a service that notifies the end-users in case of updates on the metadata of the services they are using, so they can vary the configuration and act accordingly whenever necessary. All the described toolchains are necessary, as they address already identified (listed above) tasks.

11.4.4. Missing Tools Outside the UC

N/A

11.5. Declared Tools

We proceeded to a development/setup under EdgeX to achieve some features of the management of data. Semi-Automated integrated process to add new Service Providers. GUI is fully customized.

The use case has been defined more in detail over the course of the second year and the declaration of tools given in D4.1 does not correspond to the present one as some of them have been unified (i.e. their functions were not separable).

Name	This is a tool in the following phase	Compatible with AHF	Inputs	Outputs
Measurement Services	Not a Tool	No	N/A	Data sent to EdgeX
EdgeX	Developmen t/evolution	No	Data coming from sensors	Data sent to MQTT broker
Vital-IoT	Developmen t/evolution	Yes	Data collected from MQTT Broker	Visualization of data
Robofuse	Developmen t/evolution		Data coming from sensors	Visualization of data

12. UC-08.2 [ST-I, Eurotech, IUNET, POLITO, REPLY, BEIA, ROP] SoS engineering of IoT edge devices (AI-Driven Environmental Monitoring)

Contact: Davide Brunelli [davide.brunelli@unibo.it]
The description of each Use Case can be found in D1.2

12.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 3

AHT-EPP	AI-driven Camera	AI Algorithms with vector processing for Ai-Camera	IoT integration framework
1 Requirements	StkH1 and StkH3 acquire the specification describing the features of the camera and type of identification from the customer (StkH 2). Specification documents are passed in the form of Word files.	StkH3 together with experts of StkH 1 analyse the legacy wireless availability by the customer deployment environment. Documents are done in Word files.	StkH1 contributes to the requirements for the specific wireless and energy requirements. Documents are done in Word files.
2 Functional Design	StkH 2 design and simulate the AI-Camera Hardware.	StkH 3 will develop a model of the Classification algorithm for the AI-Camera.	
3 Procurement & Engineering	StkH 2 designs the PCB with Orcad and selects the components and technologies for the PCB design. Develop the firmware for AI-Camera sensors using vector processing compiler and GCC toolchain. Request quotes from the supplier for the electronic and sensor components selected in the PCB design process. The telecommunication module is embedded on the platform for supporting the services offered by the telecommunication provider (StkH 2). A working prototype of the AI-camera will be tested.	StkH3 will implement the AI model in Python for being executed on the AI-Camera. A first release of the SW will be tested on the AI-Camera prototype working on the real condition. A Continuous Integration approach is adopted for the development of the core and following updates.	StkH2: Design and development of the IoT integration framework, using the Eclipse Java toolchain, Github, Maven, etc. Selection and acquisition of external software services (e.g. Amazon AWS) and software licenses.
4 Deployment & Commissioning	StkH1 produce the AI-Camera and supply them to the vendor product lines	StkH3 compiles and installs the program on the AI-Camera before being supplied to the StkH1 product lines.	StkH2: Deployment and pre-commissioning test of the prototypes. Deployment and commissioning of StkH1 instance of the IoT integration framework.

5 Operation & Management			The IoT framework becomes a tool used to monitor a fleet of AI-Cameras. StkH2 is in charge of monitoring the StkH1's instance of the IoT integration framework and ensures it is operating correctly.
6 Maintenance Decommissioning & Recycling			
7 Evolution	Data collected in the Operation & Management phase of StkH2 are analyzed by StkH1 for identifying possible bottlenecks in the AI-Camera design. Moreover, firmware updates will be generated by StkH3 to keep a good standard of cybersecurity.	Data collected in the Operation & Management phase of StkH2 are analyzed by StkH3 for improving the algorithm and increasing the energy performances.	StkH2: identification of potential updates of existing IoT integration framework, of new releases with significant improvements, etc. No tools currently support this phase.
8 Training & Education	StkH1 create the datasheet of AI-Camera PCB and reference manual of low-level API	StkH3 will document the code to be shared with the StkH1	The operator uses the user manual of the IoT integration framework provided by StkH2 to learn how to monitor the fleet of smart boilers. StkH3: source code is documented and specific tools are used to generate the related documentation. User manuals are edited using Microsoft Word.

12.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 4

AHT-EPP	**AI-driven Camera**	**AI Algorithms with vector processing for Ai-Camera**	**IoT integration framework**
1 Requirements	Microsoft Word: Document with the requirements defined for the Ai-Camera.	Microsoft Excel + Word: Document with the requirements elicitation.	Microsoft Excel + Word: Document with the requirements elicitation.
2 Functional Design	Matlab: design of the control algorithm to be implemented on the AI-Camera model. GitLab: SW versioning system	Python and Matlab.GitLab: SW versioning system	

3 Procurement & Engineering	Orcad: design of PCB of the AI-Camera GitLab: SW versioning system GCC compiler. GCC toolchain: framework for installing applications on the AI-Camera	Python, Matlab GitLab: SW versioning system	Eclipse IDE: Eclipse Tools for IoT platform GitHub: SW versioning system AWS: Cloud computing services
4 Deployment & Commissioning	GitLab: SW versioning system.	GitLab: SW versioning system.	Eclipse IDE: Eclipse Tools for IoT applications
5 Operation & Management			Eclipse IDE: Eclipse Tools for IoT platform IoT integration framework toolchain: Developed by StKH2 will be used for remote monitoring and management of boiler fleets. Microsoft Excel + Word: Document with reports.
6 Maintenance Decommissioning & Recycling	Microsoft Excel + Word: Document with reports.	Microsoft Excel + Word: Document with reports.	Microsoft Excel + Word: Document with reports.
7 Evolution	Microsoft Excel + Word: Document with reports.	Microsoft Excel + Word: Document with reports.	Microsoft Excel + Word: Document with reports.
8 Training & Education	Microsoft Excel + Word+Powerpoint: Document with reports.	Microsoft Excel + Word+Powerpoint: Document with reports.	Microsoft Excel + Word+Powerpoint: Document with reports.

12.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 3

AHT-EPP	**AI-driven Camera**	**AI Algorithms with vector processing for Ai-Camera**	**IoT integration framework**
1 Requirements	Microsoft Office: manual connection	Microsoft Office: manual connection	Microsoft Office: manual connection
2 Functional Design	Matlab: automatic connection with Simulink. Simulink: automatic connection with Matlab.	Python: manual GitLab: automatic	

	GitLab: automatic		
3 Procurement & Engineering	Orcad: manual GitLab: automatic connection with Orcad and GCC: manual	Python: manual GitLab: automatic	GitHub: automatic AWS: semi-automatic
4 Deployment & Commissioning	GitLab: automatic.	Python: manual GitLab: automatic	Eclipse IDE: manual
5 Operation & Management			Eclipse IDE: manual IoT integration framework toolchain: semi-automatic Microsoft Excel + Word: manual connection
6 Maintenance Decommissioning & Recycling	Microsoft Office: manual connection	Microsoft Office: manual connection	Microsoft Office: manual connection
7 Evolution	Microsoft Office: manual connection	Microsoft Office: manual connection	Microsoft Office: manual connection
8 Training & Education	Microsoft Office: manual connection	Microsoft Office: manual connection	Microsoft Office: manual connection

12.4. Gap Analysis

12.4.1. Level of automation of the toolchain & Information passed

The toolchain has an initial level of automation that will be improved during the project. Some tools can not be automated such as the production and consultation of documentation, which are MS Word + Excel + PowerPoint.

The information passed between the requirements and the functional design phases is in the form of documentation and is managed manually. The same is valid for the information passed between the functional design and the procurement/engineering phases.

The information passed between the procurement/engineering and the deployment phases is represented by a .zip file containing the entire application system to be deployed. No information is passed between the deployment and the operation phases. Finally, the engineering phase passes all source codes to the evolution phase.

12.4.2. Level of integration with the Arrowhead Framework

The toolchain is in a first stage setup of the Arrowhead Framework. The data analytics are produced for Python processing and they will be integrated automatically.

12.4.3. Missing Tools within the UC

StkH2 will implement the AI-Driven camera modules and will provide them to potential third-party consumers by using the Arrowhead Framework.

12.4.4. Missing Tools Outside the UC

N/A

12.5. Declared Tools

The AI-Camera represents an engineering tool for facilitating the deployment of easy control of accesses, even by third-party consumers.

Name	This is a tool in the following phase	Compatible with AHF	Inputs	Outputs
AI-driven Camera	<StkH1 - EPP 4,5>	Not at the moment, we are evaluating the possibility to automatize the passage of information from the operation & management of StkH1 to the Evolution phase of StkH2 by using the AHT Framework interfaced by the IoT integration framework.	Image acquired	Image identification
AI Algorithms with vector processing for Ai-Camera	<StkH2 - EPP 4, 5>	No	<StkH2 - EPP4, 5> Image data from the camera	<StkH2 - EPP4, 5> data visualization of the access control system
IoT integration framework	<StkH1 - EPP5> <StkH2 - EPP5> <StkH2 - EPP6>	Yes	<StkH2 - EPP4, 5, 6> Raw image from the AI camera	<StkH2 - EPP4, 5, 6> Processed data produced from the AI-camera

13. UC-08.3 [ST-I, Eurotech, IUNET, POLITO, REPLY, BEIA, ROP] SoS engineering of IoT edge devices (Condition Monitoring)

Contact: Federico Montori [federico.montori2@unibo.it]
The description of each Use Case can be found in D1.2

13.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 5

AHT-EPP	SN & CH	Gas Sensor	Sensor Network (WAE)	Visualizer (Configurer)	Local Cloud Gateway
1 Requirements	Excel, Project Management, Matlab executable specs	na	na	na	Requirements elicitation. Specifications definition.
2 Functional Design	Design of the sensors	Task partitioning, programming, simulation, configuration	Study of the placement of the sensors on the structure, study of how to connect each other and where to put Cluster Heads	na	Functional architecture definition.
3 Procurement & Engineering	Procurement of PCB manufacturing,	Board prototyping, Virtualization of gas metering value in an API that an IoT node/gateway can call	Acquire the necessary gateways and hardware for the Mist WoT with Public IP address. Development of the Mist WoT	Acquire the necessary gateways and hardware for the Visualizer with Public IP address. Development of the Visualizer.	During this phase, two processes develop in parallel: - design, development, test and debug of the Local Cloud Gateway hardware and software; design, development, test and debug of the use case specific business logic; - interaction with suppliers to acquire electronic and mechanical components, define external services to manufacture hardware prototypes; interaction with suppliers to acquire source code, software libraries and software licenses potentially required for the software part of the gateway. The objective is the production of the Local Cloud Gateway golden sample.
4 Deployment &	Virtual sensors fielding and	na	Linking of the Visualizer and the	Linking of the Visualizer and the Mist	Deployment of the prototypes to test and debug them in a real environment to evaluate their

Commissioning	estimation of positioning		Mist WoT with the SN	WoT with the SN	maturity. Commissioning of the golden sample to start the production of the product.
5 Operation & Management	Sensors connected to CH and therefore to the cloud	na	Usage of the Visualizer to infer potential Hazards	Infer Hazards	Integration of use case components on the field. Data collection, storage, local processing. Remote monitoring and control. Interfacing with cloud and enterprise level.
6 Maintenance Decommissioning & Recycling	Verify if something is broken	na	Configure the sensor optimally in order not to waste energy resources. Done Manually.	na	Remote monitoring and control for maintenance and decommissioning purposes.
7 Evolution	na	task partitioning, programming, simulation, configuration	Linking of the Visualizer to the Gas Sensor and PMUT as well. Manual redesign.	na	Faults and bugs analysis to identify solutions, improvements and new product releases.
8 Training & Education	na	na	na	na	Generation of design and source code documentation. Editing of technical and user manuals.

13.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 5

AHT-EPP	**SN & CH**	**Gas Sensor**	**Sensor Network (WAE)**	**Visualizer (Configurer)**	**Local Cloud Gateway**
1 Requirements	na		na	na	Currently no tools support this phase.
2 Functional Design	Matlab: language with IDE Eagle: PCB design and electrical schematic LTSpice: Circuit simulator		Matlab: language with IDE simulation tools:	na	Currently no tools support this phase.

3 Procurement & Engineering	STM32CubeIDE: IDE for programming in C Matlab: language with IDE		Visual Code: IDE for programming in Javascript Terminal: prompt in Linux Node.js: Framework for Javascript	Visual Code: IDE for programming in Javascript Terminal: prompt in Linux Node.js: Framework for Javascript	Eclipse IDE: Java development workspace extended by plugins Java Runtime Environment: runtime environment to run the Java application system Maven: build automation tool use in the software production phase Java Virtual Machine: virtual machine that enables to run Java applications GitHub: source code management, versioning and ticketing platform ZenHub: agile development form GitHub platform Docker: open platform for developing, shipping and running applications OpenShift: layered system designed to expose underlying Docker-formatted container images Prometheus: event monitoring and alert platform Zookeeper: configuration and synchronization service that coordinates the configuration process of distributed applications Linux terminal: command shell for the execution of Linux commands JavaDoc: tool for the automatic generation of HTMLpages of API documentation from Java source files
4 Deployment & Commissioning	Matlab: language with IDE		Terminal: prompt in Linux SN & CH	Terminal: prompt in Linux	Eclipse IDE: base development workspace extended by plugins Java Runtime Environment: runtime environment to run the Java application system Maven: build automation tool Java Virtual Machine: virtual machine that enables to run Java applications GitHub: source code management, ticketing and versioning platform ZenHub: agile development from GitHub platform Docker: open platform for developing, shipping and running applications OpenShift: layered system designed to expose underlying Docker-formatted container images

					Prometheus: event monitoring and alert platform Zookeeper: configuration and synchronization service that coordinates the configuration process of distributed applications Linux terminal: command shell for the execution of Linux commands Eclipse Kura: IoT framework to manage the remote gateway, collect data and process data on the edge Eclipse Kapua: edge infrastructure integration and fleet management
5 Operation & Management	Matlab: language with IDE		**Visualizer (Configurer), SN & CH**	Domain experts: It's actually people	Docker: open platform for developing, shipping and running applications OpenShift: layered system designed to expose underlying Docker-formatted container images Prometheus: event monitoring and alert platform Zookeeper: configuration and synchronization service that coordinates the configuration process of distributed applications Linux terminal: command shell for the execution of Linux commands on the multiservice gateway Eclipse Kura: IoT framework to manage the remote gateway, collect data and process data on the edge Eclipse Kapua: edge infrastructure integration and fleet management
6 Maintenance Decommissioning & Recycling	Matlab: language with IDE		**Visualizer (Configurer)**	na	Linux terminal: command shell to reach the multiservice gateway Eclipse Kura: IoT framework to manage the remote gateway, collect data and process data on the edge Eclipse Kapua: edge infrastructure integration and fleet management
7 Evolution	na		Gas Sensor PMUT Network Local Cloud Gateway	na	Currently no tools support this phase.

			Visual Code: IDE for programming in Javascript Terminal: prompt in Linux Node.js: Framework for Javascript		
8 Training & Education	na			na	JavaDoc: tool for the automatic generation of HTML pages of API documentation from Java source files ReadMe: tool to manage technical documentation

13.3.
13.4. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 5

AHT-EPP	**SN & CH**	**Gas Sensor**	**Sensor Network (WAE)**	**Visualizer (Configurer)**	**Local Cloud Gateway**
1 Requirements	na		na	na	Not applicable
2 Functional Design	Matlab: manual Eagle: manual LTSpice: manual		Matlab: manual simulation tools:	na	Not applicable
3 Procurement & Engineering	STM32CubeIDE: manual Matlab: manual		Visual Code: manual Terminal: manual Node.js: automatic	Visual Code: manual Terminal: manual Node.js: automatic	- Hardware toolchain Eclipse IDE: main integrated tool automatic the development toolchain Java Runtime Environment: automatic connection with Eclipse IDE Maven: automatic connection with Eclipse IDE Java Virtual Machine: automatic connection with Eclipse IDE

					GitHub: automatic connection with Eclipse IDE ZenHub: automatic connection with GitHub Docker: already integrated in the toolchain OpenShift: already integrated in the toolchain Prometheus: already integrated in the toolchain Zookeeper: already integrated in the toolchain Linux terminal: manual usage JavaDoc: automatic
4 Deployment & Commissioning	Matlab: manual		Terminal: manual SN & CH: manual	Terminal: manual	Gateway configuration (including Kura): manual. Deployment and commissioning, after configuration: automatic. Eclipse IDE: main integrated tool automatic the development toolchain Java Runtime Environment: automatic connection with Eclipse IDE Maven: automatic connection with Eclipse IDE Java Virtual Machine: automatic connection with Eclipse IDE GitHub: automatic connection with Eclipse IDE ZenHub: automatic connection with GitHub Docker: already integrated in the toolchain OpenShift: already integrated in the toolchain Prometheus: already integrated in the toolchain Zookeeper: already integrated in the toolchain Linux terminal: manual usage
5 Operation & Management	Matlab: manual		**Visualizer (Configurer): manual**	Domain experts: na	Eclipse IDE: main integrated tool automatic the development toolchain

			SN & CH: automatic		Java Runtime Environment: automatic connection with Eclipse IDE Maven: automatic connection with Eclipse IDE Java Virtual Machine: automatic connection with Eclipse IDE GitHub: automatic connection with Eclipse IDE ZenHub: automatic connection with GitHub Docker: already integrated in the toolchain OpenShift: already integrated in the toolchain Prometheus: already integrated in the toolchain Zookeeper: already integrated in the toolchain Linux terminal: manual usage
6 Maintenance Decommissioning & Recycling	Matlab: manual		**Visualizer (Configurer): manual** **SN & CH: manual**	na	Linux terminal: manual usage
7 Evolution	na		Gas Sensor: manual PMUT Network: manual Local Cloud Gateway: manual Visual Code: manual Terminal: manual Node.js: automatic	na	Not applicable
8 Training & Education	na			na	JavaDoc: automatic ReadMe: manual

13.5. Gap Analysis

13.5.1. Level of automation of the toolchain & Information passed

The toolchain at M0 is pretty much manual all over. During the project, we will make extensive use of the Arrowhead Framework in order to automate many of the connections. The gas sensor will be integrated with the AHF so it is automatically discovered. A new tool will take

over the Visualizer (the persister) and the Visualizer will be integrated with the AHF and be automatically connected to the sensor network as a Configurer. Regarding the Local Cloud Gateway, we plan to simplify/automate the definition, implementation and execution of simple business logics on the edge through a graphical dataflow editor, embedded in Kura. This will allow the operator, during the functional design phase, to define a business logic without writing code. During the phases 4, 5 and 6 the integration with the AF will allow the automatic publication of services (and related information), the usage of services/functionalities offered by the components of the use cases, the automatic publication and usage of cross-domain services.

The essential information passed from phase to phase is the position of the sensor nodes and their endpoints (manual for now in phases 4 and 6, while the sensor data is passed automatically in phase 5). The information and the endpoint of the gas sensor are passed manually as well in phase 7. Local Cloud Gateway:

- phase 1-2-3: requirements and specifications are passed manually through documents of spreadsheets.
- phase 3:
    - information from suppliers are managed through documents of spreadsheets.
    - source code is passed between tools automatically.
- phase 4-5-6: gateway configuration, gateway information, use case specific collected and processed data are passed between tools automatically.
- phase 7: information is managed manually.
- phase 3-8: gateway design and source code are passed to automatic documentation tools.

13.5.2. Level of integration with the Arrowhead Framework

At M0 there is no use of the AHF whatsoever. During the project we will develop and integrate several tools that will result in service providers and consumers for the AHF:

- The Gas Sensor will be a service provider
- The Persister will be a Service consumer

The WAE will act as a service provider for the whole sensor network, it will also be a consumer for the Configurer, who in turn is a provider against the WAE as it pushes configurations for sensor nodes. The Configurer will interact ultimately with the Optimizer in phases 3-4 and 6 and it is going to be a provider for it, in fact the Optimizer is expected to be a service consumer pushing the optimization configuration.

This information flow however will be implemented either in a push based way (as it is described) WAE<-Configurer<-Optimizer, or in a pull-based way, therefore roles are swapped and it is WAE->Configurer->Optimizer. This is to be planned, depending on the Arrowhead Technology (the outcome is the same anyway). The Local Cloud Gateway will be integrated with the AF. Currently, both Eclipse Kura and Kapua, on which the Gateway will be based, are not integrated with the AF. The Local Cloud Gateway will consume the services published by

the use case components on the local cloud, will offer services related to the local business logic and will bridge the local cloud to cross-domain services (both producer and consumer).

13.5.3. Missing Tools within the UC

As said previously, we identify:

- Gas Sensor (integration of a legacy tool) - Stkh2
- Optimizer (new tool) - Stkh1
- Configurer (integration of a legacy tool) - Stkh1
- Persister (new tool, built on top of Kura) - Stkh3
- WAE (new tool) - Stkh1
- Duty Cycle Management (integration of a legacy tool) - Stkh1
- Power Harvesting (new tool) - Stkh1
- Dataflow IDE - Stkh3
- Use case specific business logic (built in Kura) - Stkh3

The most essential components are within the scope of the sensor ecosystem, together with the interconnecting parts. The Optimizer is a supporting tool, its function can be replaced by a human without affecting the data flow and the behavior of the SoS (although with increased cost and processing time). Eclipse Kura and Kapua are critical for the remote management functionalities, data collect and processing, interaction with cloud platform and service provisioning.

13.5.4. Missing Tools Outside the UC

We could think of adding sensors of any kind to our ecosystem as long as they are AHF-compatible and could contribute to the SHM goals. Furthermore, we could think of possibly needing a tool that allows us to have a digital twin of the structure to be monitored (maybe a 3D design) to see if it facilitates the design of the sensor positions.

13.6. Declared Tools

During Y2 the tools were identified and the definition provided here are accurate enough to reflect what will be actually in place. D4.1 reported only an initial design which was much more coarse-grained, in fact the definition of the tools has been broken down into several components here. We have not reported here the PMUT network as its definition is in progress and needs additional investigation.

Name	This is a tool in the following phase	Compatible with AHF	Inputs	Outputs
SN & CH	Not a tool (this is the baseline group of sensors)	-	-	-

SN & CH (power harvesting)	4.5	no		
Gas Sensor	now 7, will be in 4,5,6	expected	signals from environment	Gas level, humidity and temperature
Sensor Network (WAE)	will be in 5,6,7	expected	Inertial sensors and their WoT endpoints	AHT sensor proxies and their endpoints
Visualizer (Configurer)	now 5, will be in 4,5,6	expected	optimized configuration	optimized configuration in WoT format
Optimization	will be in 2,4,6	expected	position and number of inertial sensors, historian of gas level	optimized duty cycle
Persister	will be in 5,6,7	expected	SHM inertial data, gas sensor data, PMUT data	visualization of data values
Local Cloud Gateway	will be in 4,5,6	expected	Raw data from the sensors	Raw and processed data produced from the sensors and published as a service

14. UC-08.4 [ST-I, Eurotech, IUNET, POLITO, REPLY, BEIA, ROP] SoS engineering of IoT edge devices (Smart Home)

Contact: Davide Brunelli [davide.brunelli@unibo.it], Edoardo Patti [edoardo.patti@polito.it], Gianvito Urgese [gianvito.urgese@polito.it], Sara Bocchio [sara.bocchio@st.com]
The description of each Use Case can be found in D1.2

14.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 4

AHT-EPP	Smart Gas Meter	Smart Electric meter (hw existent)	LF-NILM	IoT integration platform
1 Requirements	StkH 2 acquire the specification describing the electromechanical design of the SBS from the vendor (StkH 1). Mechanical parts, sensors and actuators are described with reference to the datasheets. Specification documents are passed in the form of PDF files that are analyzed by the engineering team that generate a list of requirements for the design of the smart meter. Requirements will be listed in a Word file.	StkH 2 update the specification according to the evolution outcome	StkH4 discusses with the owner of the legacy infrastructure what kind of energy data can be exposed and how those data can be consumed by the system. Therefore, the requirements of the system are defined in accordance with the decisions of both parties. All system requirements will be clearly recorder by using MS Word+Excel+PowerPoint. Most of documentation will be saved in a PDF format and eventually consulted by using a PDF reader such as Adobe Acrobat Reader.	Elicitation of the requirements; definition of the specifications
2 Functional Design	StkH 2 design and simulate the system model of the meter/transmission algorithm of the Smart meter in matlab. Design of the chip and the board are following traditional EDA design kit	StkH 2 update the design and simulate the system model of the meter/transmission algorithm of the Smart meter in matlab. Design of the chip and the board are following traditional EDA design kit	StkH4 designs the system modules in accordance with the requirements defined in the previous phase.	The functional design is performed according to the guidelines given in the "Requirements" phase
3 Procurement & Engineering	StkH2 produce the Smart meter and supply them to the vendor product lines	The updated firmware from evolution is deployed to the different installations of the smart meters (if required) or deployed in a new generation of smart meter	StkH4 receives from the owner of the legacy infrastructure the credentials to access the energy data stored in the database. StkH4 develops the system modules (provider and consumer) and the	- Supply chain planning based on functional design aiming at an efficient and cost-effective procurement (acquire the software and hardware goods needed from suppliers to build the prototype)

			web dashboards for data visualization. StkH4 develops a security mechanism to ensure that only authorized third-party consumers can access energy data.	- Development of the IoT integration platform by using a well defined toolchain; - Design and development of the use case specific business logic - test of the IoT integration platform
4 Deployment & Commissioning		The updated firmware from evolution is deployed to the different installations of the smart meters (if required) or deployed in a new generation of smart meter	StkH4 mounts the provider module on a Docker container running on a server. The consumer modules are delivered from time to time to third parties interested in using the energy data supplied by the legacy infrastructure.	Deployment of the prototype and test in a real environment. At the end of the test, setup and deployment of the components of the use case, StkH3 is involved in the commissioning of the complete solution, focusing on the IoT integration platform.
5 Operation & Management			The system is up and running: the provider disseminates the energy data, while the consumers use them.	The multiservice gateway allows data collection, edge processing and remote management of the metering infrastructure
6 Maintenance Decommissioning & Recycling				Remote identification of malfunctions is facilitated by remote management. Criticalities prediction is enabled by constant monitoring of the infrastructure status
7 Evolution	Data collected in the Operation & Management phase of StkH 1 are analyzed by StkH2 for identifying possible bottlenecks in the smart meter design.	Data collected in the Operation & Management phase of StkH 1 are analyzed by StkH2 for identifying possible bottlenecks in the smart meter design. Moreover, firmware updates will be generated by StkH2 for update/extend communication standards	StkH4 may plan to extend the system with additional services from Python data analytics.	Faults and bug analysis to identify solutions, improvements and new product releases.
8 Training & Education			StkH4 produces the training material for both end-users and developers.	Generation of source code documentation. Editing of technical and user manuals.

14.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 4

AHT-EPP	Smart Gas Meter	Smart Electric meter (hw existent)	LF-NILM	IoT integration platform
1 Requirements	Acrobat Reader: Documents are in PDF format analyzed manually. Microsoft Word: Document with the requirements defined for the ECU.		MS Word: requirements documentation MS Excel: tabular information for requirements elicitation MS PowerPoint: graphical presentations for requirements elicitation Adobe Acrobat Reader: consult PDF files generated during requirements elicitation	Currently no tools support this phase
2 Functional Design	Matlab: design of the control algorithm to be implemented on the smart meter model. GitLab: SW versioning system EDA RTL simulation, synthesis for hardware design gcc toolchain: firmware development	GCC toolchain: framework for update the firmware on the Smart meter	MS Word: functional design documentation MS Excel: tabular information of functional design MS PowerPoint: graphical contents of functional design Adobe Acrobat Reader: consult PDF files generated during functional design	Currently no tools support this phase
3 Procurement & Engineering	Orcad: design of PCB of the Smart meter GCC toolchain: framework for installing applications on the Smart meter	GCC toolchain: framework for installing applications on the Smart meter	Java SE Development Kit 13: development environment for building the Java application system Java Runtime Environment: runtime environment to run the Java application system Spring Framework: comprehensive framework for building web applications based on Java SpringToolSuite4: IDE for developing Spring-based applications Maven: tool for automating the building of the application system MS SQL Server: relational database management system adopted by the legacy infrastructure to store energy data MS Java Database Connectivity driver (JDBC): driver for connecting the Java	Eclipse IDE: Java development workspace extended by plugins Java Runtime Environment: runtime environment to run the Java application system Maven: build automation tool use in the software production phase Java Virtual Machine: virtual machine that enables to run Java applications GitHub: source code management and versioning platform ZenDesk: agile development from GitHub platform Docker: open platform for developing, shipping and running applications OpenShift: layered system designed to expose underlying Docker-formatted container images

			application system to the database of the legacy infrastructure HTML+CSS3+JavaScript: front-end programming languages for developing the consumer dashboard Bootstrap: CSS framework to develop responsive front-end web pages Python: programming language for producing data analytics from energy data Jupyter Notebook: open-source web application for interactive computing in the Python programming language MS Word+Excel+PowerPoint: tools for documenting the engineering process Adobe Acrobat Reader: for consulting PDF files	Prometheus: event monitoring and alert platform Zookeeper: configuration and synchronization service that coordinates the configuration process of distributed applications Linux terminal: command shell for the execution of Linux commands JavaDoc: tool for the automatic generation of HTMLpages of API documentation from Java source files
4 Deployment & Commissioning			Docker: virtualization platform for deploying the provider module on a server Java Runtime Environment: runtime environment to run the Java application system MS SQL Server: relational database management system adopted by the legacy infrastructure to store energy data HTML+CSS3+JavaScript: front-end programming languages for the consumer dashboard	Eclipse IDE: base development workspace extended by plugins Java Runtime Environment: runtime environment to run the Java application system Maven: build automation tool Java Virtual Machine: virtual machine that enables to run Java applications GitHub: source code management, ticketing and versioning platform ZenDesk: agile development form GitHub platform Docker: open platform for developing, shipping and running applications OpenShift: layered system designed to expose underlying Docker-formatted container images Prometheus: event monitoring and alert platform Zookeeper: configuration and synchronization service that coordinates the configuration process of distributed applications Linux terminal: command shell for the execution of Linux commands Eclipse Kura: IoT framework to manage the remote gateway,

				collect data and process data on the edge Elicpse Kapua: edge infrastructure integration and fleet management
5 Operation & Management			Docker: virtualization platform for deploying the provider module on a server Java Runtime Environment: runtime environment to run the Java application system MS SQL Server: relational database management system adopted by the legacy infrastructure to store energy data HTML+CSS3+JavaScript: front-end programming languages for the consumer dashboard	Docker: open platform for developing, shipping and running applications OpenShift: layered system designed to expose underlying Docker-formatted container images Prometheus: event monitoring and alert platform Zookeeper: configuration and synchronization service that coordinates the configuration process of distributed applications Linux terminal: command shell for the execution of Linux commands on the multiservice gateway Eclipse Kura: IoT framework to manage the remote gateway, collect data and process data on the edge Eclipse Kapua: edge infrastructure integration and fleet management
6 Maintenance Decommissioning & Recycling				Linux terminal: command shell to reach the multiservice gateway Eclipse Kura: IoT framework to manage the remote gateway, collect data and process data on the edge Eclipse Kapua: edge infrastructure integration and fleet management
7 Evolution	Acrobat Reader: Documents are in PDF format analyzed manually. Microsoft Word: Document with the requirements defined for the smart meter		MS Word+Excel+PowerPoint: tools for documenting the requirements of planned evolution Adobe Acrobat Reader: for consulting PDF files	Currently no tools support this phase
8 Training & Education			MS Word+Excel+PowerPoint: tools for creating training material Adobe Acrobat Reader: for consulting PDF files	JavaDoc: tool for the automatic generation of HTML pages of API documentation from Java source files

				ReadMe: tool to manage technical documentation

14.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 4

AHT-EPP	**Smart Gas Meter**	**Smart Electric meter (hw existent)**	**LF-NILM**	**IoT integration platform**
1 Requirements	Acrobat Reader: Manual Microsoft Word: Manual		MS Word: manual connection MS Excel: manual connection MS PowerPoint: manual connection Adobe Acrobat Reader: manual connection	not applicable
2 Functional Design	Matlab: Manual GitLab: SW versioning system EDA RTL simulation, Manual gcc toolchain: Manual	GCC toolchain: Manual connection	MS Word: manual connection MS Excel: manual connection MS PowerPoint: manual connection Adobe Acrobat Reader: manual connection	not applicable
3 Procurement & Engineering	Orcad: Manual GCC toolchain: Manual	GCC toolchain: Semiautomatic	Java SE Development Kit 13: automatic connection with SpringToolSuite4 Java Runtime Environment: automatic connection with SpringToolSuite4 Spring Framework: automatic connection with SpringToolSuite4 SpringToolSuite4: automatic connection with SpringFramework, JDK13, JRE, Maven Maven: automatic connection with SpringToolSuite4 MS SQL Server: automatic connection with JDBC MS Java Database Connectivity driver (JDBC):	Eclipse IDE: main integrated tool automatic the development toolchain Java Runtime Environment: automatic connection with Eclipse IDE Maven: automatic connection with Eclipse IDE Java Virtual Machine: automatic connection with Eclipse IDE GitHub: automatic connection with Eclipse IDE Docker: already integrated in the toolchain OpenShift: already integrated in the toolchain Prometheus: already integrated in the toolchain Zookeeper: already integrated in the toolchain

			semi-automatic connection with SpringToolSuite4 HTML+CSS3+JavaScript: semi-automatic connection with SpringToolSuite4 Bootstrap: semi-automatic connection with SpringToolSuite4 Python: automatic connection with MS SQL Server Jupyter Notebook: automatic connection with Python MS Word+Excel+PowerPoint: manual Adobe Acrobat Reader: manual	Linux terminal: manual usage JavaDoc: automatic
4 Deployment & Commissioning			Docker: automatic connection with JRE Java Runtime Environment: automatic connection with Docker, MS SQL Server and HTML+CSS3+JavaScript MS SQL Server: automatic connection with JRE HTML+CSS3+JavaScript: automatic connection with JRE	Eclipse IDE: main integrated tool automatic the development toolchain Java Runtime Environment: automatic connection with Eclipse IDE Maven: automatic connection with Eclipse IDE Java Virtual Machine: automatic connection with Eclipse IDE GitHub: automatic connection with Eclipse IDE Docker: already integrated in the toolchain OpenShift: already integrated in the toolchain Prometheus: already integrated in the toolchain Zookeeper: already integrated in the toolchain Linux terminal: manual usage
5 Operation & Management			Docker: automatic connection with JRE Java Runtime Environment: automatic connection with Docker, MS SQL Server and HTML+CSS3+JavaScript MS SQL Server: automatic connection with JRE HTML+CSS3+JavaScript: automatic connection with JRE	Docker: automatic connection OpenShift: automatic connection Prometheus: automatic connection Zookeeper: automatic connection Linux terminal: manual connection

6 Maintenance Decommissioning & Recycling				Linux terminal: manual usage
7 Evolution	Acrobat Reader: Manual Microsoft Word: Manual	Acrobat Reader: Manual Microsoft Word: Manual	MS Word+Excel+PowerPoint: manual Adobe Acrobat Reader: manual	not applicable
8 Training & Education			MS Word+Excel+PowerPoint: manual Adobe Acrobat Reader: manual	JavaDoc: automatic ReadMe: manual

14.4. Gap Analysis

14.4.1. Level of automation of the toolchain & Information passed

LF-NILM: The toolchain already has a good level of automation, since the different tools are supported by the same frameworks and development environments supporting the Java programming language. Furthermore, the provider and the consumer modules of the application system are provided with the Arrowhead Framework Client Skeleton, thus guaranteeing a good automation level between the tools composing the system. The only tools that can not be automated are those involving the production and consultation of documentation, which are MS Word + Excel + PowerPoint and Adobe Acrobat Reader. The information passed between the requirements and the functional design phases is in the form of documentation and is managed manually. The same is valid for the information passed between the functional design and the procurement/engineering phases. The information passed between the procurement/engineering and the deployment phases is represented by a JAR file containing the entire application system to be deployed. No information is passed between the deployment and the operation phases. Finally, the engineering phase passes all source codes to the evolution phase.

IoT integration platform: The toolchain adopted is composed of two classes of tools. The former is an integrated and automated toolchain that is used for code development and which comprises JVM, JRE, Maven, Eclipse IDE, JavaDoc and Github. ReadMe, also used for documentation preparation, has a lower degree of automation. One of the goals of the project is to tackle this gap and enable ReadMe to keep the documentation up-to-date by adding the description of issues found in the source code. These tools are intended for a software product preparation and do not become part of the final IoT platform. The second set of tools (Docker, OpenShift, Prometheus, Zookeeper) are components of the final IoT platform and their level of integration and automation has already reached a high level of maturity.

- phase 1-2 → 3: requirements and specifications are passed manually through documents and spreadsheets.
- phase 3:
  - information from suppliers is managed through documents and spreadsheets.
  - source code is passed between tools automatically.
  - hardware design? TBD
- phase 4-5-6: gateway configuration, gateway information, use case specific collected and processed data are passed between tools automatically.
- phase 7: information is managed manually.
- phase 3-8: gateway design and source code are passed to automatic documentation tools.

14.4.2. Level of integration with the Arrowhead Framework

The toolchain is well integrated with the Arrowhead Framework, because the application system is an extension of the Arrowhead Client Skeleton based on Java Spring. The additional tools that are not provided with the Client Skeleton are automatically integrated with the Arrowhead Framework once they properly included in the application, which is itself based on the Arrowhead Client. The additional tools to be integrated are the MS SQL Server, the MS JDBC for SQL server and the tools for developing the web dashboard (Bootstrap, HTML, CSS3 and JavaScript). The data analytics produced in Python will also be automatically integrated once inserted in the database of the legacy infrastructure.

14.4.3. Missing Tools within the UC

StkH1 will implement the LF-NILM component that will consume energy disaggregation data from an existing legacy infrastructure and will provide them to potential third-party consumers by using the Arrowhead Framework. The MS SQL Server, as any other possible SQL or NoSQL database management system, is crucial for the LF-NILM integration, since the whole system is based on an already deployed legacy infrastructure.

14.4.4. Missing Tools Outside the UC

N/A

14.5. Declared Tools

The Use Case also involves the development of a High Frequency NILM for which, compared to D4.1, there are no updates. It is briefly described as follows and added consequently to the table below. The home high-frequency NILM software executes the task of disaggregation power consumption of single appliances from an agglomerated mains power measurement. From the machine learning point of view, this is considered a single-channel blind source separation problem, where multiple sources need to be extracted from one combined measurement. The NILM core algorithms are developed for embedded ultra-low power Vector processors and use algorithms tested and validated on a Matlab project. The training phase is still on server side (SQL/NoSQL dB), while classification is expected real-time on the meter.

Name	**This is a tool in the following phase**	**Compatible with AHF**	**Inputs**	**Outputs**
LF-NILM	<StkH4 - EPP4, 5>	Yes, since the system is an extension of the Arrowhead Client Skeleton implemented in Spring.	<StkH4 - EPP4, 5> energy data from the database of the legacy infrastructure	<StkH4 - EPP4, 5> data visualization of available energy data by the consumers through the use of dashboards
IoT integration platform	<StkH3 - EPP4, 5, 6>	expected	<StkH3 - EPP4, 5, 6> Raw data from the metering infrastructure	<StkH3 - EPP4, 5, 6> Raw and processed data produced from the metering infrastructure and published as a service
HF-NILM	EPP 2, 7	expected	Matlab	C++/C#

15. UC-08.5 [ST-I, Eurotech, IUNET, POLITO, REPLY, BEIA, ROP] SoS engineering of IoT edge devices (Industrial Energy Monitoring)

Contact: Antonio Lionetto [antonio.lionetto@st.com]
The description of each Use Case can be found in D1.2

There are no updates from this Use Cases in comparison with D4.1.
For conformity, we report here the content adapted to the D4.2 format.

Gap Analysis

Condition monitoring and anomaly detection is the process of monitoring relevant physical characteristics such as vibration, noise, temperature, power adsorption of an equipment and, as part of this, to recognize the trend versus an anomalous behavior. Advanced monitoring is based on the appropriate set of sensors in which sensor fusion capability reinforces the status detection and providing the required data collection enable their analysis through the different vertical layer providing advanced services on base of this information and elaboration chain.
This industrial scenario consists of:

- Smart sensor nodes deployed at the critical machines
- Collection of sensors data and preprocessing with/without machine learning at the edge (either the node or a gateway / industrial PC)
- High-level analytics on company premises or on cloud for provisioning the nodes with updated algorithms

Today the applications enabled by this approach based on smart sensor vibration and ultrasound nodes are growing, providing an advanced monitoring on production line and ensuring a condition maintenance for critical machineries to avoid downtime.

15.1. Declared Tools

Name	This is a tool in the following phase	Compatible with AHF	Inputs	Outputs
Autonomous sensing enabling kit	EPP2, EPP7	no	TBD	TBD
Edge side processing	EPP2, EPP7	no	TBD	TBD
Debugging tool	EPP5	expected	TBD	TBD

16. UC-09 [Acciona, dotGIS, AITIA, BME]: Machine operation optimisation

Contact: José Luis Burón [joseluis.buron.martinez@acciona.com]
The description of each Use Case can be found in D1.2

There are no updates from this Use Cases in comparison with D4.1.
For conformity, we report here the content adapted to the D4.2 format.

16.1. Gap Analysis

The main objective is to track the operations of the machinery in real time in order to monitor the progress of the earthworks, detect deviations from the original plan, analyze productivity and key performance indicators, and detect opportunities for optimization of the operations.
It would be interesting to explore ways of automating the interaction between the different phases of the UC-EP, as the current interactions are rather manual.

16.2. Declared Tools

We identify here 4 tools:

- Tool for automating the link between product backlog and tasks defined in the Functional Design phase.
- Tool for modeling the platform architecture and for the management of the specifications of the different platform modules. Currently, separate specification documents (Word, PowerPoint and/or Excel files) are created for each module and stored in a folder of the repository of Microsoft teams. Therefore, what is lacking is a tool for a structured management of the platform architecture and of the specifications of the different modules.
- Tool for automating the Quality Control stage in the Engineering Phase, facilitating the definition and management of tests for verifying the software modules produced in the Development stage of the Engineering Phase.
- Tools for automating the Documentation stage of the Engineering Phase, especially for software modules.

Name	This is a tool in the following phase	Compatible with AHF	Inputs	Outputs
Automating Tool	EPP2	no	TBD	TBD
Modeling Tool	EPP2	no	TBD	TBD
Quality Control	EPP3	no	TBD	TBD

Documentation	EPP8	no	TBD	TBD

17. UC-10 [ARCELIK, EDI, UTIA]: Rapid HW development, prototyping, testing and evaluation

Contact: Mustafa Küçükkuru [mustafa.kucukkuru@arcelik.com], Alper Özel [alper.ozel@arcelik.com], Çağlar Henden [caglar.henden@arcelik.com]

The description of each Use Case can be found in D1.2

17.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 6
No tools used in phase 5

AHT-EPP	Embedded Zynq Ultrascale+ Unit (ZynqU+)	FMC A/D (A/D)	Relays(Relay)	FPGA Control Unit for Power/Load (FPGA)	Remote reconfiguration of FPGA (RemCtrl)	Accelerated digital design on multiple PCs (ParDesign)
1 Requirements	StkH 2 acquires the specification describing the measurement system from the vendor (StkH 1). Specification documents are passed in the form of PDF files that are analyzed by the engineering team that generates a list of requirements for the design of the ZynqU+. Requirements will be listed in a Word file.	StkH2 will analyze existing design templates provided by company Analog devices for Xilinx University boards.	StkH1 will specify relay system needed for switching of analog measurement points	StkH 4 acquires the specification describing the FPGA from the vendor (StkH 1). Specification documents are passed in the form of PDF files that are analyzed by the engineering team that generates a list of requirements for the design of the FPGA. Requirements will be listed in a Word file.	StkH 4 acquires the specification describing the RemCtrl requirements from the vendor (StkH 1). Specification documents are passed in the form of PDF files that are analyzed by the engineering team that generates a list of requirements for the design of the FPGA RemCtrl. Requirements will be listed in a Word file.	StkH2 will acquire specification for maximal HW design compilation times related to modifications and upgrades ZynqU+ programmable logic designs for single PC installation and requirements for their acceleration on several PCs from StkH1. Specification documents are passed in the form of PDF files that are analyzed by the engineering team that generates a list of requirements for the ParDesign of the ZynqU+. Requirements will be listed in a Word file.
2 Functional Design	StkH 2 will design and simulate the HW part of the ZynqU+ interfaces in Xilinx Vivado simulation tool xsim.	StkH2 will model design templates ported to selected ZynqU+ module and carrier and model it in Xilinx Vivado simulation tool xsim.	StkH1 will model in Matlab isolation properties of the Relay in respect to isolation of high voltage environment	StkH 4 will design and simulate the HW part of the FPGA interfaces in Modelsim.	StkH 4 will design and simulate the remote communication part of the RemCtrl interfaces in Matlab.	StkH 2 will design and simulate the parallel compilation HW part of the ZynqU+ interfaces in Matlab parallel toolbox.

3 Procurement & Engineering	StkH 2 will specify the off-the-shelf for the ZynqU+ module and carrier PCBs . Develop the firmware for Arm and related PL logic interfaces to A/D using Xilinx GCC and Vivado toolchains. Request quotes from the supplier for the electronic and sensor components selected in the PCB design process. The engineering team will develop the documentation of the HW and SW developed for the ZynqU+ that will be provided as a reference manual to the StkH 1. A working prototype of the ZynqU+ will be tested by StkH1 in the vendor laboratories.	StkH1 and Stk2 will buy identical A/D data acquisition card for the Zynq U+ system	StkH 1 will design the Relay PCB with Altium Designer and select the components and technologies for the PCB design. Develop the firmware for drive sensors and actuators using STM32 GCC toolchain for STM32 Nucleo controller subsystem.	StkH 4 will specify the off-the-shelf for the FPGA system module and carrier PCB . StkH5 will develop the firmware for MicroBlaze softcore and HW interfaces to RS232 serial interfaces using Xilinx GCC and Vivado toolchains. Request quotes from the supplier for the electronic and sensor components selected in the PCB design process. The ethernet is embedded on the platform for supporting the services offered by the telecommunication provider (StkH 4). The engineering team will develop the documentation of the HW and SW developed for the FPGA that will be provided as a reference manual to the StkH 1. A working prototype of the FPGA will be tested in a real SBS machine for quality testing in the vendor laboratories.	StkH 4 will specify the off-the-shelf for the RemCtrl for FPGA. StkH5 will develop the firmware for MicroBlaze softcore and HW interfaces to RS232 serial interfaces using Xilinx GCC and Vivado toolchains. Request quotes from the supplier for the electronic and sensor components selected in the PCB design process. The ethernet is embedded on the platform for supporting the services offered by the telecommunication provider (StkH 4). The engineering team will develop the documentation of the HW and SW developed for the FPGA that will be provided as a reference manual to the StkH 1. A working prototype of the FPGA will be tested in a real SBS machine for quality testing in the vendor laboratories.	StkH2 will specify used design tool sequence and define Pardesign compilation schedules acceleration on several PCs.
4 Deployment & Commissioning	StkH2 and Stkh3 produce the ZynqU+ SW firmware and HW design and supply them to the vendor StkH1. StkH 3 will utilize embedded AH framework SW C++ clients running on the	StkH2 produces the ZynqU+ SW firmware and HW design to control A/D and supply them to the vendor StkH1.	StkH1 will produce the Relay SW firmware and HW design and supply them to the vendor StkH2	StkH4 and StkH5 produce the FPGA SW firmware for MicroBlaze and HW design and supply them to the vendor StkH1	StkH4 and StkH5 produce the RemCtrl SW and HW and supply them to the vendor StkH1	StkH 2 will utilize AH framework for control of execution of tools in case of use of ParDesign.

	embedded Debian Linux for communication of the ZynqU+ with remote control/visualization PC clients.		and StkH5.			
6 Maintenance Decommissioning & Recycling						
7 Evolution	Data collected in the Operation & Management phase of StkH 1 are analyzed by StkH2 for identifying possible bottlenecks in the ZynqU+ design. Moreover, firmware updates and HW design updates will be generated by StkH2 as needed by experiments.	StkH2 produces the Linux version of ZynqU+ SW firmware and HW design to control A/D and supply them to the vendor StkH1.	StkH1 will produce the Relay SW firmware and HW design updates and supply them to the vendor StkH2 and StkH5.	Data collected in the Operation & Management phase of StkH 1 are analyzed by StkH5 for identifying possible bottlenecks in the FPGA design. Moreover, firmware updates and HW design updates will be generated by StkH5 as needed by experiments.	Data collected in the Operation & Management phase of StkH 1 are analyzed by StkH4 for identifying possible bottlenecks in the RemCtrl SW/HW. Moreover, firmware updates and HW design updates for RemCtrl will be generated by StkH4 and StkH5 as needed by experiments.	Data related to recompilation of HW collected in the Operation & Management phase of StkH 1 will be analyzed by StkH2 for identifying possible bottlenecks in the ParDesign. Moreover, Arrowhead firmware updates related to accelerated design scheduling will be generated by StkH2 as needed by experiments.
8 Training & Education	StkH2 and StkH3 will create the datasheet of ZynqU+ system and reference manual of low-level SW API for A53 CPU running Debian embedded Linux OS.	StkH2 creates the datasheet of A/D integration in ZynqU+ and reference manual of low-level SW API for A53 CPU running Debian embedded Linux OS.	StkH1 creates the datasheet of Relay and reference manual of low-level SW API for STM32 MPU.	StkH4 and StkH5 will create the datasheet of FPGA system and reference manual of low-level SW API for MicroBlazs soft core running in FPGA.	StkH4 and StkH5 will create the datasheet of RemCtrl system and reference manual of low-level communication protocol description.	StkH3 will create the datasheet of Design acceleration flow and reference manual describing configuration of AH framework for ParDesign acceleration of multiple PCs in a local cloud for StkH 1.

17.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 6
No tools used in phase 5

AHT-EPP	Embedded Zynq Ultrascale+ Unit (ZynqU+)	FMC A/D (A/D)	Relays(Relay)	FPGA Control Unit for Power/Load (FPGA)	Remote reconfiguration of FPGA (RemCtrl)	Accelerated digital design on multiple PCs (ParDesign)
1 Requirements	Acrobat Reader: Documents are in PDF format analyzed manually. Microsoft Word: Document with the requirements defined for the ZynqU+.	Acrobat Reader: Documents are in PDF format analyzed manually. Microsoft Word: Document with the requirements defined for the A/D	Acrobat Reader: Documents are in PDF format analyzed manually. Microsoft Word: Document with the requirements defined for the Relay.	Acrobat Reader: Documents are in PDF format analyzed manually. Microsoft Word: Document with the requirements defined for the FPGA.	Acrobat Reader: Documents are in PDF format analyzed manually. Microsoft Word: Document with the requirements defined for the RemCtrl	Acrobat Reader: Documents are in PDF format analyzed manually. Microsoft Word: Document with the requirements defined for the ParDesign.
2 Functional Design	Matlab: interpreted m scripts will call compiled C MEX-functions , Simulink: interpreted simulink blocks will call compiled S-functions, HW design simulator of Xilinx Vivado: will be used for simulation of HW design at the RTL level. Xilinx System Generator for DSP: it will be used to model Simulink bit-exact equivalents of HDL HW designs.	Matlab: interpreted m scripts will call compiled C MEX-functions , Simulink: interpreted simulink blocks will call compiled S-functions, HW design simulator of Xilinx Vivado: will be used for simulation of HW design at the RTL level. Xilinx System Generator for DSP: it will be used to model Simulink bit-exact equivalents of HDL HW designs. Make files developed and provided by company Analog Devices for interfacing of A/D cards with FMC connector standard: Make files describe HW design compilation process in Xilinx Vivado tool. Ubuntu installation of Xilinx Vivado is required.	Matlab: for modeling of supported states of the switching network.	Matlab: interpreted m scripts will call compiled C MEX-functions , Simulink: interpreted simulink blocks will call compiled S-functions, HW design simulator of Xilinx Vivado: will be used for simulation of HW design at the RTL level. Xilinx System Generator for DSP: it will be used to model Simulink bit-exact equivalents of HDL HW designs.	Matlab: interpreted m scripts will call compiled C MEX-functions , Simulink: interpreted simulink blocks will call compiled S-functions, HW design simulator of Xilinx Vivado: will be used for simulation of HW design at the RTL level. Xilinx System Generator for DSP: it will be used to model Simulink bit-exact equivalents of HDL HW designs.	Matlab: interpreted m scripts will call compiled C MEX-functions , Simulink: interpreted simulink blocks will call compiled S-functions, HW design simulator of Xilinx Vivado: will be used for simulation of HW design at the RTL level. Xilinx System Generator for DSP: it will be used to model Simulink bit-exact equivalents of HDL HW designs. Make files developed and provided by company Analog Devices for interfacing of A/D cards with FMC connector standard: Make files describe HW design compilation process in Xilinx Vivado tool. Ubuntu installation of Xilinx Vivado is required. AH tools clients: for remote control of simulations on separate PCs with Debian installations of Xilinx Vivado.

3 Procurement & Engineering	Xilinx Vivado: for compilation of HW designs from the HDL source code, Xilinx SDK: Eclipse editor with GCC SW compilers for Arm A53 processors. Xilinx Petalinux configuration scripts: for configuration of Linux kernel and device tree which reflects the custom HW present in the programmable logic part of the ZynqU+. Xilinx SDSoC compiler: Eclipse editor with Xilinx system level compiler serving for compilation of selected C/C++ SW functions into HW accelerators together with HW data movers.	Xilinx Vivado: for compilation of HW designs from the HDL source code, Xilinx SDK: Eclipse editor with GCC SW compilers for Arm A53 processors. Xilinx Petalinux configuration scripts: for configuration of Linux kernel and device tree which reflects the custom HW present in the programmable logic part of the ZynqU+. Xilinx SDSoC compiler: Eclipse editor with Xilinx system level compiler serving for compilation of selected C/C++ SW functions into HW accelerators together with HW data movers.	Altium Designer: for design of the Relay PCB.	Xilinx Vivado: for compilation of HW designs from the HDL source code, Xilinx SDK: Eclipse editor with GCC SW compilers for MicroBlaze processor.	Xilinx Vivado: for compilation of HW designs from the HDL source code, Xilinx SDK: Eclipse editor with GCC SW compilers for MicroBlaze processor.	Xilinx Vivado: for compilation of HW designs from the HDL source code, Xilinx SDK: Eclipse editor with GCC SW compilers for Arm A53 processors. Xilinx Petalinux configuration scripts: for configuration of Linux kernel and device tree which reflects the custom HW present in the programmable logic part of the ZynqU+. Xilinx SDSoC compiler: Eclipse editor with Xilinx system level compiler serving for compilation of selected C/C++ SW functions into HW accelerators together with HW data movers.
4 Deployment & Commissioning	Xilinx Vivado HLS, Xilinx SDK, Xilinx Petalinux configuration scripts, Xilinx SDSoC compiler, Xilinx Vitis acceleration flow. Arrowhead framework C++ producer services and consumers clients: for connection of ZynqU+ SW apps to remote SW clients.	Xilinx Vivado HLS, Xilinx SDK, Xilinx Petalinux configuration scripts, Xilinx SDSoC compiler, Xilinx Vitis acceleration flow. Arrowhead framework C++ producer services and consumers clients: for connection of ZynqU+ SW apps to remote SW clients.	Altium Designer: for preparation of fabrication data files for of the Relay PCB.	Xilinx Vivado HLS, Xilinx SDK, Xilinx Petalinux configuration scripts, Xilinx SDSoC compiler, Xilinx Vitis acceleration flow. Arrowhead framework C++ producer services and consumers clients: for connection of ZynqU+ SW apps to remote SW clients.	Xilinx Vivado HLS, Xilinx SDK, Arrowhead framework C++ producer services and consumers clients: for connection of ZynqU+ SW apps to remote SW clients.	Xilinx Vivado HLS, Xilinx SDK, Xilinx Petalinux configuration scripts, Xilinx SDSoC compiler, Xilinx Vitis acceleration flow. Arrowhead framework C++ producer services and consumers clients: for sequencing of file transfers and execution of tools on several PCs connected by ethernet in a local cloud for acceleration and management of system HW/SW compilation and configuration projects and managements of resulting design files.

6 Maintenance Decommissioning & Recycling	Acrobat Reader: analysis of PDF reports generated by StkH1.	Acrobat Reader: analysis of PDF reports generated by StkH1.	Altium Designer: for engineering changes of the Relay PCB if needed.	Acrobat Reader: analysis of PDF reports generated by StkH1.	Acrobat Reader: analysis of PDF reports generated by StkH1.	Acrobat Reader: analysis of PDF reports generated by StkH1.
7 Evolution	Xilinx Vivado HLS, Xilinx SDK, Xilinx Petalinux configuration scripts, Xilinx SDSoC compiler, Arrowhead framework C++ producer services and consumers clients: for connection of ZynqU+ SW apps to remote SW clients.	Xilinx Vivado HLS, Xilinx SDK, Xilinx Petalinux configuration scripts, Xilinx SDSoC compiler.	Altium Designer: for updates of schematic sources, PCB layouts and fabrication data files for the Relay PCB in case of need for redesign.	Xilinx Vivado: for compilation of FPGA HW. Xilinx SDK for compilation of MicroBlaze SW.	Xilinx Vivado: for compilation of FPGA HW. Xilinx SDK for compilation of MicroBlaze SW.	Xilinx Vivado HLS, Xilinx SDK, Xilinx Petalinux configuration scripts, Xilinx SDSoC compiler: for recompilation of HW/SW design evolutions. Arrowhead framework C++ producer services and consumers clients: for sequencing of file transfers and execution of tools on several PCs
8 Training & Education	Matlab report generator and Simulink report generator: for creation and updates of technical documentation. GitLab: SW versioning system. SVN: versioning system.	Matlab report generator and Simulink report generator: for creation and updates of technical documentation. GitLab: SW versioning system. SVN: versioning system.	Altium Designer documentation generator: for creation and updates of technical documentation. GitLab: SW versioning system. SVN: versioning system.	Matlab report generator and Simulink report generator: for creation and updates of technical documentation. GitLab: SW versioning system. SVN: versioning system.	Matlab report generator and Simulink report generator: for creation and updates of technical documentation. GitLab: SW versioning system. SVN: versioning system.	Matlab report generator and Simulink report generator: for creation and updates of technical documentation. GitLab: SW versioning system. SVN: versioning system.

17.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 6
No tools used in phase 5

AHT-EPP	**Embedded Zynq Ultrascale+ Unit (ZynqU+)**	**FMC A/D (A/D)**	**Relays(Relay)**	**FPGA Control Unit for Power/Load (FPGA)**	**Remote reconfiguration of FPGA (RemCtrl)**	**Accelerated digital design on multiple PCs (ParDesign)**
1 Requirements	Acrobat Reader: manual connection	Acrobat Reader: manual connection	Acrobat Reader: manual connection	Acrobat Reader: manual connection	Acrobat Reader: manual connection	Acrobat Reader: manual connection Microsoft Word: manual connection

	Microsoft Word: manual connection	Microsoft Word: manual connection	Microsoft Word: manual connection	Microsoft Word: manual connection	Microsoft Word: manual connection	
2 Functional Design	Matlab:manual, Simulink:manual, HW design simulator of Xilinx Vivado:script based control	Matlab, Simulink, HW design simulator of Xilinx Vivado	Matlab: manual	Matlab, Simulink, HW design simulator of Xilinx Vivado	Matlab, Simulink, HW design simulator of Xilinx Vivado	Matlab, Simulink, HW design simulator of Xilinx Vivado
3 Procurement & Engineering	Xilinx Vivado HLS:script based control, Xilinx SDK: manual, Xilinx Petalinux: configuration scripts, Xilinx SDSoC compiler: manual, Xilinx Vitis acceleration flow: manual	Xilinx Vivado HLS: script based control, Xilinx SDK: manual, Xilinx Petalinux: configuration scripts, Xilinx SDSoC compiler: manual, Xilinx Vitis acceleration flow:manual	Altium Designer:manual	Xilinx Vivado HLS:script based control, Xilinx SDK: manual, Xilinx Petalinux: configuration scripts, Xilinx SDSoC compiler: manual, Xilinx Vitis acceleration flow:manual	Xilinx Vivado HLS:script based control, Xilinx SDK: manual, Xilinx Petalinux: configuration scripts, Xilinx SDSoC compiler: manual, Xilinx Vitis acceleration flow:manual	Xilinx Vivado HLS:script based control, Xilinx SDK: script based control, Xilinx Petalinux: configuration scripts, Xilinx SDSoC compiler: script based control, Xilinx Vitis acceleration script based control, ArrowHead framework: script based automation
4 Deployment & Commissioning	Xilinx Vivado HLS: script-based control, Xilinx SDK: manual, Xilinx Petalinux: configuration scripts, Xilinx SDSoC compiler: manual. Arrowhead framework C++ producer services and consumers clients: make script based configuration and compilation of connection of ZynqU+ SW apps to remote SW clients.	Xilinx Vivado HLS, Xilinx SDK, Xilinx Petalinux configuration scripts, Xilinx SDSoC compiler: manual	Altium Designer: manual	Xilinx Vivado HLS, Xilinx SDK, Xilinx Petalinux configuration scripts, Xilinx SDSoC compiler, Xilinx Vitis acceleration flow	Xilinx Vivado HLS, Xilinx SDK, Xilinx Petalinux configuration scripts, Xilinx SDSoC compiler, Xilinx Vitis acceleration flow	Xilinx Vivado HLS: script based control, Xilinx SDK: script based control, Xilinx Petalinux: configuration scripts, Xilinx SDSoC compiler: script based control, Xilinx Vitis acceleration script based control, ArrowHead framework: script based automation
6 Maintenance Decommissioning & Recycling	Acrobat Reader: analysis of PDF reports generated by StkH1.	Acrobat Reader: analysis of PDF reports generated by StkH1.	Altium Designer: manual	Acrobat Reader: analysis of PDF reports generated by StkH1.	Acrobat Reader: analysis of PDF reports generated by StkH1.	Acrobat Reader: analysis of PDF reports generated by StkH1.
7 Evolution	Xilinx Vivado HLS: script-based control, Xilinx SDK:	Xilinx Vivado HLS, Xilinx SDK, Xilinx Petalinux	Altium Designer: manual	Xilinx Vivado HLS, Xilinx SDK, Xilinx Petalinux	Xilinx Vivado HLS, Xilinx SDK, Xilinx Petalinux configuration	Xilinx Vivado HLS: script based control, Xilinx SDK: script based control, Xilinx

	manual, Xilinx Petalinux: configuration scripts, Xilinx SDSoC compiler: manual. Arrowhead framework C++ producer services and consumers clients: make script based configuration and compilation of connection of ZynqU+ SW apps to remote SW clients.	configuration scripts, Xilinx SDSoC compiler: manual.		configuration scripts, Xilinx SDSoC compiler, Xilinx Vitis acceleration flow	scripts, Xilinx SDSoC compiler, Xilinx Vitis acceleration flow	Petalinux: configuration scripts, Xilinx SDSoC compiler: script based control, Xilinx Vitis acceleration script based control, ArrowHead framework: script based automation
8 Training & Education	Matlab report generator and Simulink report generator: manual. GitLab: SW versioning system: manual	Matlab report generator and Simulink report generator: manual. GitLab: SW versioning system: manual	Matlab report generator and Simulink report generator: manual. GitLab: SW versioning system: manual	Matlab report generator and Simulink report generator: manual. GitLab: SW versioning system:manual	Matlab report generator and Simulink report generator: manual. GitLab: SW versioning system: manual	Matlab report generator and Simulink report generator: manual. GitLab: SW versioning system: manual

17.4.

17.5. Gap Analysis

17.5.1. Level of automation of the toolchain & Information passed

The Xilinx Vivado HLS toolchain is Eclipse-based and integrates HW design flow and SW design flow for fixed selection of Zynq Ultrascale+ devices present on the university boards: ZCU102, ZCU104 and ZCU106. The existing design flow does not provide unified configuration and project-bring-up scripts to address industrial modules fitted with variable sizes of Zynq Ultrascale devices. These scripts will be provided by UTIA.

The existing design flow does not manage different platform configuration like FMC data acquisition board A or board B, with different requirements for the HW IP interfaces. UTIA will provide a unified set of configuration and project-bring-up scripts to address these variations for the Zynq Ultrascale devices on industrial modules.

At the existing level of implementation, EDI TestBed does not support any FPGA based systems. We are planning to add the support later in the project. The interaction between the engineering phases is assumed to be fully automated. Operators will only select the test configuration setup file that was pre-defined by the administrator. An authorized operator will be connected to an EVT tool via authentication.

17.5.2. Level of integration with the Arrowhead Framework

The information is passed directly to the Arrowhead core services, local cloud, database or through an appropriate API.

17.5.3. Missing Tools within the UC

Missing tools are the DAQ system for data collecting from power supply units, peripheral communication system for setup configuration and a UI to manage and operate the systems. Currently the EDI TestBed and the DAQ system of the EVT tool and its peripherals are under development status.

17.5.4. Missing Tools outside the UC

N/A

17.6. Declared Tools

Operator UI for system configuration and management, Data storage and visualization tool for test result evaluation, automated report production tool for test result reporting, High-Speed DAQ tool for data collecting and peripheral control tool for applying test configuration to peripheral devices.

The AH framework will support these tools in the design phases.

Xilinx Vivado 2018.2 design tool serves for generation of the HW design supporting the data acquisition card in HW. Xilinx Petalinux 2018.2 tool serves for configuration of Petalinux kernel for Xilinx Ultrascale+ device with user defined custom HW. Xilinx SDK 2018.2 tool serves for generation of SW board support packages. Xilinx SDK 2018.2 tool serves also for generation of user defined application SW. EDI TestBed workstation provide remote access to hardware.

The Xilinx Vivado HLS toolchain lacks the flexibility of targeting a complete family of industrial Zynq Ultrascale+ modules with a single set of configuration and project-bring-up scripts. Automation of the execution of the sequence of Xilinx tools by AH framework will add the opportunity to perform multiple potentially very long HW compilation steps on several computers in the local cloud for several HW configurations. Automation by AH tools will also remove errors related to the manual execution of the transfers of files from tool to tool.

Name	This is a tool in the following phase	Compatible with AHF	Inputs	Outputs
Embedded Zynq Ultrascale+ Unit (ZynqU+)	Initial HW design and SW for A53 is tested and works	Yes, the AH-f 4.1.3 C/C+ clients can be executed on ZynqU+	Data from A/D FMC card	Data in DDR4 user space of Linux app
FMC A/D (A/D)	Initial HW design and SW for A53 is tested and works. Vivado: controlled by tcl scripts which can be called by AH-f services. Petalinux configuration/compilation: controlled ny configuration scripts and design archive from Vivado	Yes, the AH-f 4.1.3 C/C+ clients can be executed on ZynqU+	Analog data sampled by 1 Gsample/s from Relay	Data for HW DMA data mover in the PL part of the ZynqU+

Relays(Relay)	Schematics and PCB is in design phase (Altum Designer)	No at this stage	Analog high voltage data from power-supply under test	Analog low voltage data to A/D card
FPGA Control Unit for Power/Load (FPGA)	FPGA unit is in system specification phase	No at this stage	Bitstream	Configured FPGA
Remote reconfiguration of FPGA (RemCtrl)	RemCtrl unit is in system specification phase	No at this stage	Data with bitstream sent from remote site	reconfiguration bitstream to FPGA
Digital Design on multiple PCs (ParDesign)	Design on single PC by manually executed sequence of HW design steps followed by Linux configuration and SW compilation steps.	No at this stage, but AH framework will be used for synchronization and scheduling of compilation steps on several PCs in a local cloud	HW Design source code, HW Ips, Linux configuration SW source code	SD card with BOOT.BIN with FSBL, U-BOOT, PetaLinux kernel, Debian file system, bitstream for configuration of the programmable logic part of the ZynqU+ and compiled SW application with shared libraries serving as SW abstraction of the data communication to the HW in the programmable logic.

18. UC-11 [DAC, GUT, CISC]: Configuration tool for autonomous provisioning of local clouds

Contact: Marek Tatara, Jakub Rewieński, Ralph Weissnegger, Christian Ettinger [marek.tatara@dac.digital, jakub.rewienski@pg.edu.pl, r.weissnegger@cisc.at, c.ettinger@cisc.at]

The description of each Use Case can be found in D1.2

18.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 6

AHT-EPP	Arrowhead-compliant small-footprint producer node (PN)	Field gateway (FG)	Cloud management infrastructure (CMI)	Performance assessment tool (PAT; not available at M0)	Onboarding application (OA; not available at M0)	Arrowhead Local Cloud wireless security enhancement tool
1 Requirements						
2 Functional Design						
3 Procurement & Engineering						
4 Deployment & Commissioning	When preparing a package, the producer node should be onboarded upfront to particular gateways using the onboarding application.	Field gateway should be deployed on each truck by StkH3 prior to the delivery. The envisioned onboarding process is designed to be highly automatized and simple so that a technician of StkH3 should be able to easily deploy new Field gateways as needed. Field gateway will host Eclipse Arrowhead as an interoperability framework.	When new field gateways are being deployed they are added to the database as available for further IoT nodes deployment.			This tool can be used when onboarding new devices to include spatial data to the verification of the devices to be onboarded. It might be defined as e.g. maximum distance of the sensor from the corresponding field gateway in order to admit the node to the cloud.

5 Operation & Management	In the operation phase the IoT nodes serve as measurement providers.	Filed gateway will be used as a local cloud provider and a proxy between the nodes and cloud management infrastructure.	Cloud management infrastructure gives the ability for all stakeholders to monitor the current status of the delivery, e.g. measured temperature or acceleration. Also, dynamic reconfiguration in terms of sensors configuration (like sampling time) could be dynamically changed.			Security enhancement tool can be used to constantly verify the spatial information about devices connected to the local cloud and generate events (e.g. suggest deauthentication) if a device leaves a defined area.
6 Maintenance Decommissioning & Recycling	Reconfiguration of the system might be needed when it's operating, which should be possible from the cloud management infrastructure level.			This tool will be installed on a field gateway to provide the information about the performance of local cloud in terms of e.g. CPU usage, link quality.		
7 Evolution	When extending the local clouds new nodes could be attached by scanning through NFC credentials of the node. Once the node is onboarded and then powered up, it pairs with an appropriate field gateway and starts providing measurements.	Field gateway implements automagic discovery procedure for paring nearby IoT nodes that were previously onboarded to this gateway.	When onboarding a new device, this tool passes the information about the deployed local clouds to the onboarding application, and once the device is being onboarded - passes the data to the selected field gateways to provide credentials of the nodes that should be admitted to the local cloud.		Credentials of the onboarded IoT node are scanned through NFC and the node is configured (e.g. type of sensor attached, field gateways to which the node should be admitted). The configuration is passed to the cloud management tool.	
8 Training & Education						

18.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 6

AHT-EPP	Arrowhead-compliant small-footprint producer node (PN) development toolchain	Field gateway (FG) development toolchain	Cloud management infrastructure (CMI) development toolchain	Performance assessment tool (PAT; not available at M0) development toolchain	Onboarding application (OA; not available at M0) development toolchain	Arrowhead Local Cloud wireless security enhancement tool
1 Requirements	Google Meet Google docs		GitLab		GitLab	MS Teams MS SharePoint
2 Functional Design	Google docs GitLab		GitLab	GitLab	GitLab	GitLab
3 Procurement & Engineering	GitLab STM32Cube IDE GCC toolchain		IntelliJ IDEA Docker	IntelliJ IDEA Docker	Android Studio	GitLab Visual Studio Code GCC toolchain
4 Deployment & Commissioning	STM32Cube IDE GCC toolchain		Docker Kubernetes	Docker Kubernetes	Flutter	GCC toolchain Python
5 Operation & Management	FG	Terminal + SSH CMI	Kubernetes Rancher	Kubernetes Rancher		React & NodeJS
6 Maintenance Decommissioning & Recycling			Rancher Prometheus	JMeter Java Runtime Environment:		
7 Evolution	GitLab STM32Cube IDE GCC toolchain				Android Studio	GitLab
8 Training & Education	GitLab		GitLab	GitLab	GitLab	GitLab

18.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 6

AHT-EPP	Arrowhead-compliant small-footprint producer node (PN) development	Field gateway (FG) development toolchain	Cloud management infrastructure (CMI)	Performance assessment tool (PAT; not	Onboarding application (OA; not available at M0)	Arrowhead Local Cloud wireless security enhancement

	toolchain		development toolchain	available at M0) development toolchain	development toolchain	tool
1 Requirements						
2 Functional Design						
3 Procurement & Engineering						
4 Deployment & Commissioning	At M0 the deployment of nodes requires a manual connection and configuration of each node by a qualified technician	A script for the configuration and deployment of Arrowhead core services has been developed to automate the process of deploying new local clouds. A process for the deployment of new field gateways in field should be prepared in order to register field gateways in the cloud management infrastructure. Here, manual plugging in and configuration of a field gateway is required.				Currently deployment of the nodes is done fully manually. Nodes have to be configured and connected to the network and to the Arrowhead Local Cloud.
5 Operation & Management			Reconfiguration of nodes is always triggered by the human operator as needed.			
6 Maintenance Decommissioning & Recycling				The tool provides the performance indicators automatically.		

7 Evolution	New node should be scanned through the onboarding application through NFC. Once it is onboarded, it should be manually powered on, and pairing with the field gateway should be handled automatically.		When onboarding new nodes the configuration data are passed to the field gateways automatically.		The onboarded node should be scanned through NFC, and then configured in the application. After that, the configuration should be sent to the cloud management infrastructure.	
8 Training & Education						

18.4. Gap Analysis

18.4.1. Level of automation of the toolchain & Information passed

At M0 the IoT nodes have to be configured and integrated manually, without automation of this process. Moreover, each node has to be integrated manually by a qualified system integrator. Therefore, the automation of deployment of new nodes is the desired functionality that will reduce the engineering costs of the system integrator, and through the cloud management infrastructure, remote diagnostics of the solution (both nodes and gateway) is possible. Moreover, due to the developed onboarding procedure, the system is intended to be easily deployable, so even the SH3 should be able to deploy it seamlessly itself.

The only automation at M0 is connected with passing the data from a node, through a gateway to the cloud management infrastructure.

For the use case execution (M36)

It is assumed that CMI is set up and new gateways can be deployed in field and will be registered in the database attributed to a particular user using the OA, where the public key of particular FG to the CMI remains, which is required to authenticate the FG. After that, new IoT nodes can be onboarded (using the onboarding toolchain) to particular FG as follows:

- a new PN is scanned by OA reading its public key
- the node is configured in the application (e.g., FG to which the node should be admitted is selected)
- the configuration data is passed from OA to CMI
- CMI passes the configuration data to a particular FG
- FG waits for powering on the node and then pairs with PN
- FG passes the configuration to the PN
- PN starts providing measurements do FG
- FG passes measurements to CMI
- CMI visualizes the data obtained from sensors

If Wireless Security Enhancement Tool is used, Smart Antenna can discover the node and after the spatial information is confirmed, it will be provided with network credentials and authenticated with Arrowhead Local Cloud.

18.4.2. Level of integration with the Arrowhead Framework

At M0 there is no integration with Arrowhead.

At M36 Arrowhead is intended to be hosted at the Field Gateway, which will be the integration platform for the wireless sensor network. All IoT nodes will be Arrowhead-compliant and will be systems with services producing measurement data and services consuming configuration data.

18.4.3. Missing Tools within the UC

All of the above-mentioned tools being a part of the UC have to be redesigned and developed from scratch in order to achieve the desired efficiency and compliance with the Arrowhead Framework. That applies to the whole onboarding toolchain, Security Enhancement Tool and Performance Assessment Tool.

The most critical components are Field Gateway and IoT Provider Node. This is the core functionality that shows the operation of an envisioned measurement system. The onboarding process (i.e. Onboarding Application and Cloud Management Infrastructure), Security Enhancement Tool and Performance Assessment Tool are not indispensable, but are the means to achieve the desired reduction of the engineering costs at a certain, around 40%-50% level.

18.4.4. Missing Tools Outside the UC

In the toolchain the system modeling tool or arrowhead management tool can be integrated into.

18.5. Declared Tools

Name of the tool	This is a tool in the following phase:	Compatible with AHF	Inputs	Outputs
Arrowhead-compliant small-footprint producer node (PN)	4 Deployment & Commissioning 5 Operation & Management 7 Evolution	Yes	Configuration of the node	Measurement data Credentials of the node
Field gateway (FG)	4 Deployment & Commissioning 5 Operation & Management 7 Evolution	Yes		

Cloud management infrastructure (CMI)	4 Deployment & Commissioning 5 Operation & Management 6 Maintenance, Decommissioning & Recycling 7 Evolution	No	Configuration of the onboarded node	
Performance assessment tool (PAT; not available at M0)	6 Maintenance, Decommissioning & Recycling	Yes		Performance indicators for a local cloud
Onboarding application (OA; not available at M0)	4 Deployment & Commissioning 7 Evolution	No	Credentials of the onboarded node	Configuration of the onboarded node
Arrowhead Local Cloud wireless security enhancement tool	4 Deployment & Commissioning 5 Operation & Management	Yes	Connecting Node MAC or ID. Wireless signal from connecting Node (to calculate the spatial information)	Spatial information about the Node. Network configuration for the node.

19. UC-12 [SAP, Jotne, Tellu, NTNU, HIOF]: Digital Twins and structural monitoring

Contact: Guoyuan Li [guoyuan.li@ntnu.no]

The description of each Use Case can be found in D1.2

19.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 5

We have negotiated with crane suppliers for sensor installation proposal; contact crane operator for expertise information about operation and maintenance; tested sensor data transmission to TelluCloud through a bike ride experiment; installed the sensors on the crane; creating 3D crane models; creating crane dynamic model;…

AHT-EPP	Sensor investigation	Sensor installation	crane modeling and simulation	data transmission	crane behavior and structural monitoring
1 Requirements	Had several meetings with crane supplier; discussed possible sensors that can be installed on the crane;	Sensor shipping; time schedule for both crane supplier and the available time for NTNU Gunnerus ship; electrical technician;	Crane 3D models used for visualization; crane dynamic model used for crane dynamic simulation and control; Finite element analysis;	IoT sensor data transmission from crane to remote end;	Need sensor data to feed into the system; replay the crane operation based on sensor data; analysis crane operation safety; analysis crane structure during operation;
2 Functional Design	Tested sensor data transmission with TelluCloud; determined sensor type and installation method;	Determined shipping time and installation time	Crane 3D models from NX; crane dynamic model from 20sim		
3 Procurement & Engineering	Got a quoto; and accept the quoto	Received sensor and relevant components;	Obtain rough crane 3D model from crane supplier; refined the 3D model; developed crane dynamic model in 20sim;		
4 Deployment & Commissioning		Installed sensors on the crane; has used wrong type of Pluto Gateway; need to replace it and update PLC;			
5 Operation & Management					
6 Maintenance Decommissioning & Recycling					

7 Evolution					
8 Training & Education					

19.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 5

AHT-EPP	**Sensor investigation**	**Sensor installation**	**crane modeling and simulation**	**data transmission**	**crane behavior and structural monitoring**
1 Requirements	**Excel**: collect sensor information				
2 Functional Design	**Excel**: collect sensor information		**NX:** 3D model building. **FEDEM**: finite element analysis. **20sim:** create crane dynamic	**TelluCloud**: IoT data transmission.	
3 Procurement & Engineering	**TelluCloud:** sensor data transmission in the bike experiment. **Jotne EMD:** data visualization and storage in the bike experiment.	**Pluto Manager:** software to manage Pluto gateway.	**NX:** 3D model building. **FEDEM**: finite element analysis **20sim:** create crane dynamic		
4 Deployment & Commissioning					
5 Operation & Management					
6 Maintenance Decommissioning & Recycling					
7 Evolution					
8 Training & Education					

19.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 5

No automation has been implemented yet.

19.4. Gap Analysis

19.4.1. Level of automation of the toolchain & Information passed

We are now using FEDEM, NX and 20sim for modeling. The toolchain is not formed yet for the use case. We consider the information passed between the 3D models and the dynamic models can be operation data and on-board sensor data.

19.4.2. Level of integration with the Arrowhead Framework

We are now working on integrating the TellU solution for data transmission into AHT. From the experience of bike ride experiment, we have succeeded in transmitting data via AHT. Now we encounter a new challenge that a Pluto gateway is deployed at the crane side. How to communicate between TellU gateway and Pluto Gateway

19.4.3. Missing Tools within the UC

A data transmission tool is needed in the use case. Now the requirement has set up, and TellU and Jotne are working on it.

19.4.4. Missing Tools Outside the UC

N/A

19.5. Declared Tools

Name of the tool	This is a tool in the following phase:	Compatible with AHF	Inputs	Outputs
Sensor investigation	Not a tool	no	use case requirements	sensor types, installation limitations
Sensor installation	Not a tool	No	Sensor types, installation limitations	determine install method and success to install sensor on the crane
crane modeling and simulation	EPP3	No	rough crane model from crane supplier	refined crane model
data transmission	EPP3	Yes, we succeeded to achieve data transmission in the bike ride experiment. Regarding the crane use case, there is a slight difference of network configuration. We are working on it.	Sensor data, Tellu gateway and cloud service	receive sensor data from Jotne EMD system
crane behavior and structural monitoring	Not start yet			

20. UC-13 [Boliden, LTU, BnearIT]: Deployment engine for production related sensor data

Contact: Markus Frank [markus.frank@boliden.com]

The description of each Use Case can be found in D1.2

20.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 3

StkH1 (SH1) => Data destination responsible / requestor
StkH2 (SH2) => SIB team
StkH3 (SH3) => Data source responsible

AHT-EPP	SIB deployment	SIB adapter	SIB data exchange
1 Requirements	SH1 requirement for data exchange in a local cloud, e.g. production site where no SIB instance is in place	SH1 requirement for data exchange where for either source or destination an adapter is missing, SH3 involved if source system is affected	SH1 requirement for new data needed in local or global cloud, this also includes transformation/translation requirements potentially needed. Other integration metadata is capture such as frequency, security and other service level related requirements
2 Functional Design	N/A	SH2 design adaptation for either source or destination system	SH2 designs the translation/transformation in alignment with SH1 and SH3
3 Procurement & Engineering	Internal SIB team plans and prepare deployment	Internal SIB team plans and prepare deployment	SH2 implements provisioned data
4 Deployment & Commissioning	Internal SIB team plans and prepare deployment	Internal SIB team plans and prepare deployment	SH2 implements and SH1,SH3 test fulfillment in delivery according to specification
5 Operation & Management	Handled by SH2	Handled by SH2	handled by SH2
6 Maintenance Decommissioning & Recycling	Handled by SH2	Handled by SH2	handled by SH2. Regular review for usage performed
7 Evolution	Handled by SH2 in alignment with evolution of SIB components and feedback from all SH1 and SH3 (all instances)	Handled by SH2 in alignment with evolution of SIB components and feedback from all SH1 and SH3 (all instances)	Handled by SH2 in alignment with feedback from SH3 and SH1 (for changes coming from data sources)
8 Training & Education	SH2 creates needed documentation for instance deployment	SH2 creates needed documentation for instance deployment	SH2 educated data users on data usage possibility, limitation and duties

20.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 3

AHT-EPP	SIB deployment	SIB adapter	SIB data exchange
1 Requirements	Microsoft office suite	Microsoft office suite	Microsoft office suite
2 Functional Design	Microsoft office suite	Microsoft office suite	Microsoft office suite
3 Procurement & Engineering	Microsoft office suite, Github	Microsoft office suite, Github	Microsoft office suite, Github
4 Deployment & Commissioning	Github, PM2 process manager, NodeJS, MQ, MQTT, Datastore SQL/noSQL	Github, PM2 process manager, NodeJS, MQ, MQTT, Datastore SQL/noSQL	Github, PM2 process manager, NodeJS, MQ, MQTT, Datastore SQL/noSQL
5 Operation & Management	Github, PM2 process manager, NodeJS, MQ, MQTT, Datastore SQL/noSQL	Github, PM2 process manager, NodeJS, MQ, MQTT, Datastore SQL/noSQL	Github, PM2 process manager, NodeJS, MQ, MQTT, Datastore SQL/noSQL
6 Maintenance Decommissioning & Recycling	Github, PM2 process manager, NodeJS, MQ, MQTT, Datastore SQL/noSQL	Github, PM2 process manager, NodeJS, MQ, MQTT, Datastore SQL/noSQL	Github, PM2 process manager, NodeJS, MQ, MQTT, Datastore SQL/noSQL
7 Evolution	Microsoft office suite,	Microsoft office suite,	Microsoft office suite,
8 Training & Education	Microsoft office suite, DokuWiki, Github	Microsoft office suite, DokuWiki, Github	Microsoft office suite, DokuWiki, Github

20.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 3

AHT-EPP	SIB deployment	SIB adapter	SIB data exchange
1 Requirements	Manual based on standards	Manual based on standards	Manual based on standards
2 Functional Design	Manual based on standards, Github related automated	Manual based on standards, Github related automated	Manual based on standards, Github related automated
3 Procurement & Engineering	Manual based on standards, Github related automated	Manual based on standards, Github related automated	Manual based on standards, Github related automated
4 Deployment & Commissioning	Manual based on standards, Github related automated	Manual based on standards, Github related automated	Manual based on standards, Github related automated
5 Operation & Management	Manual based on standards, Github related	Manual based on standards, Github related automated	Manual based on standards, Github related automated

	automated		
6 Maintenance Decommissioning & Recycling	Manual based on standards, Github related automated	Manual based on standards, Github related automated	Manual based on standards, Github related automated
7 Evolution	Manual based on standards, Github related automated	Manual based on standards, Github related automated	Manual based on standards, Github related automated
8 Training & Education	Manual based on standards, Github related automated	Manual based on standards, Github related automated	Manual based on standards, Github related automated

20.4. Gap Analysis

20.4.1. Level of automation of the toolchain & Information passed

With the legacy approach no standardization, reusability and automation was possible and if vendor approaches are used the integration is dependent on the vendors capability. The SIB provides a simple and fast scalable approach to consolidate new production related data sources into historians or to other data receivers. The integration in the toolchain is still moderate with focus on deployment and commissioning but adds standardization into the process.

Information passed from the requirement phase is source and destination system specification, transformation/ translation requirements as well as quality and security requirements and relevant contacts. This includes also the assessment of a newly needed adaptor if not already present.

From functional design to procurement & engineering the technical specifications for potentially new adaptations are needed as well as translation/transformation specifications. The SIB teams manage the implementation and deployment and commissioning and uses the implemented adapters, translation specs and technical information of source and destination systems to implement the data exchanges after internal testing and from the destination system responsible. Information passed to operation & management is the documentation of above-mentioned information into the SIB wiki, which is also used in phase 6. For evolution information comes from feedback of source or destination system owner, events, incident and problems of the SIB as well as technology development of use of open source components. All information is documented in the SIB wiki and also used for training & education.

20.4.2. Level of integration with the Arrowhead Framework

As the integration is primarily on data level (interoperability level) the interaction with the arrowhead framework can be with gateway services as a bridge to other local or global clouds.

20.4.3. Missing Tools within the UC

SIB is available for use for other partners within the AHT EU project.

LTU is investigating virtualization in the edge, which matches the requirements for “local survivability” and the production requirements of high availability and scalability.

20.4.4. Missing Tools Outside the UC

N/A

20.5. Declared Tools

Name of the tool	This is a tool in the following phase:	Compatible with AHF	Inputs	Outputs
Simple integration box	Deployment & Commissioning, Operation & Management, Maintenance Decommissioning & Recycling	on interoperability level, compatible with Gateways system	Source data	Destination data after translation

21. UC-14 [IFAG, IFAT]: Smart Diagnostic Environment for Contactless Module Testers

Contact: Christian Hanser [christian.hanser@infineon.com], Georg Skacel [georg.skacel@infineon.com]

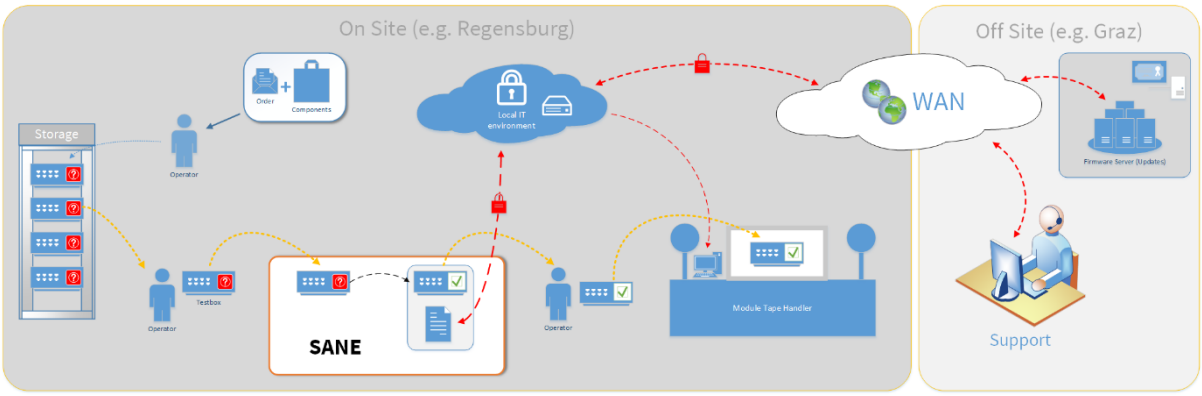
The description of each Use Case can be found in D1.2

The use case assumptions were not changed since Y1.

21.1. Gap Analysis

21.2.

Currently, however, the FADE tool lacks support for remote diagnosis and remote maintenance. Thus, right now on-site expert support is needed every time an error occurs. Moreover, downtimes to search failures as well as additional measurement cycles (as the same tape must be re-measured after a repair) result in high costs. So does regular maintenance - mainly due to maintenance contracts and high shipping costs, as servicing is currently only done at the IFAT premises in Graz. The new SANE tool will improve upon the FADE tool by building a smart remote environment for the accurate, step-wise smart diagnosis and maintenance of ISO-module test boxes.



21.3. Declared Tools

Name	This is a tool in the following phase	Compatible with AHF	Inputs	Outputs
SANE	Operation & Management, Maintenance	No	sensor data from the boxes	reports of failures and accurate diagnosis

22. UC-15 [VTC, LTU, BnearIT] Virtual Commissioning of a Cyber-Physical System for increased flexibility

Contact: Richard Hedman [richard.hedman@volvo.com]

22.1. Use Case Overview

Kitting is to ensure material is delivered and presented to the operator in an optimal way. It gives the opportunity of optimizing material flow and to balance the workload efficiently and effectively. With the introduction of electromobility and automation, our assembly lines and material flow will be even more complex than today. The work of this use case consists in developing concepts for smart and automated kitting operations.

22.2. Identified Gaps

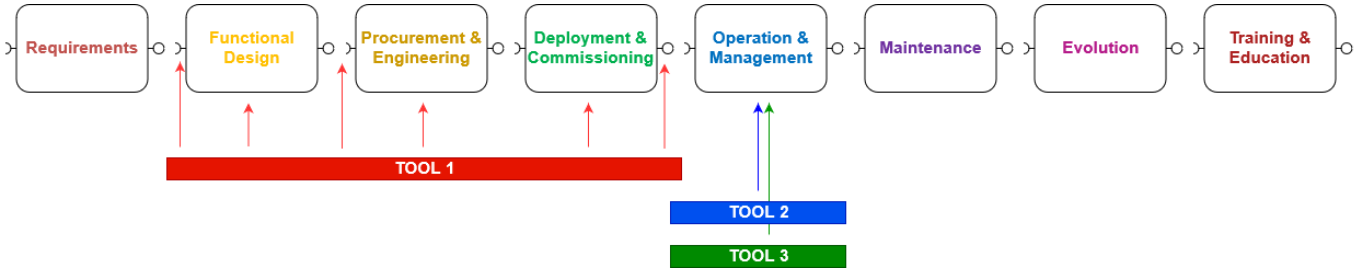
Today, the automation in the kitting processes mainly concern operator support, i.e. pick to light. There is little automation in terms of process planning and monitoring and control of kitting operations. Also, there is a low level of interoperability between concerned systems, such as MES and the logistics planning systems. Initially, the use case will focus on the engineering process related process preparation for kitting. It involves actions of the roles of Logistics Engineer and Production Engineer, aiming for virtual preparation of components, tools and work instructions.

22.3. Identified Needed Tools

Tool 1. MPMLink (Windchill) – used for adaptive control over the kitting wagons and
   a. Design-time tool
   b. Connected to the AHF
   c. EPP2, EPP3, EPP4
   d. Inputs: DCN [EP-I2, EP-I3]
   e. Outputs: [definitions what and how to assemble] [EP-O4]

Tool 2. Central Planning Office - used to define products that have to be assembled
   a. Run-time tool
   b. Connected to the AHF
   c. EPP5
   d. Inputs: -
   e. Outputs: [order specification]

Tool 3. Manufacturing Execution System - used for supervision of the manufacturing process progress
   a. Run-time tool
   b. Connected to the AHF
   c. EPP5
   d. Inputs: -
   e. Outputs: [order progress]


Requirements
Functional Design
Procurement & Engineering
Deployment & Commissioning
Operation & Management
Maintenance
Evolution
Training & Education
TOOL 1
TOOL 2
TOOL 3

23. UC-15 [VTC, LTU, BnearIT] Virtual Commissioning of a Cyber-Physical System for increased flexibility

Contact: Richard Hedman [richard.hedman@volvo.com]

The description of each Use Case can be found in D1.2

23.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 1

AHT-EPP	**NOT PROVIDED**
1 Requirements	
2 Functional Design	
3 Procurement & Engineering	
4 Deployment & Commissioning	
5 Operation & Management	
6 Maintenance Decommissioning & Recycling	
7 Evolution	
8 Training & Education	

23.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 1

AHT-EPP	**NOT PROVIDED**
1 Requirements	
2 Functional Design	
3 Procurement & Engineering	
4 Deployment & Commissioning	
5 Operation & Management	
6 Maintenance Decommissioning & Recycling	
7 Evolution	
8 Training & Education	

23.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 1

AHT-EPP	**NOT PROVIDED**
1 Requirements	
2 Functional Design	
3 Procurement & Engineering	

4 Deployment & Commissioning	
5 Operation & Management	
6 Maintenance Decommissioning & Recycling	
7 Evolution	
8 Training & Education	

23.4. Gap Analysis

23.4.1. Level of automation of the toolchain & Information passed

23.4.2. Level of integration with the Arrowhead Framework

23.4.3. Missing Tools within the UC

23.4.4. Missing Tools Outside the UC

23.5. Declared Tools

NOT PROVIDED

Name	**This is a tool in the following phase**	**Compatible with AHF**	**Inputs**	**Outputs**

24. UC-16.1 [Bosch]: Energy efficiency

Contact: Johannes Kristan, Michael Leippert, Valentin Fetscher [Valentin.Fetscher@de.bosch.com]

The description of each Use Case can be found in D1.2. This is a new use case.

24.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 4

AHT-EPP	Gateway Software Components	Hardware Components	Backend Software Components	Equipment Integration (EI)
1 Requirements	N/A	Define physical signal(s) to measure and the required sampling rate for specific use case. (Current@0.5kHz-1kHz, Voltage@1Hz)	Requirements are collected in Excel sheets and detailed in accompanied word and power point documents and tracked In issue tracker (Atlassian JIRA). As some of the components are open source, additional issues are created in external issue tracking systems (github.com) to facilitate development there.	N/A
2 Functional Design	Check sensors for supported data formats. Convert format of data to required format. (JSON, CSV) Aggregation of data	Selecting sensor(s) able to measure required physical signals with the required frequency. (Advantech WISE-4010, PicoScope XXXX) Select Hardware for processing and conversion of data formats. (Kunbus RevPi)	For the functional design word documents as well as diagrams in UML and non-standardized notations are used.	N/A
3 Procurement & Engineering	N/A	Hardware is supplied by external partner specific to use case	Procurement is done with SAP and internal procurement systems. The engineering for the backend is done with usual software development tooling such as an integrated development environment (InteliJ IDEA), revision control systems (git, stash), builds are automated with Jenkins build server.	N/A
4 Deployment & Commissioning	Software on the PoC is deployed by external partner using their standard process. (SSH, VPN)	Configure static DHCP for devices.	ssh, ftp, kubernetes, kubectl	Webservice deployed in EI environment for communication with AH environment.
5 Operation & Management	TBD	TBD	Top, syslog, kubectl, prometheus	TBD

6 Maintenance Decommissioning & Recycling	Maintenance for PoC done by engineering and external partners.	Maintenance for PoC done by engineering and external partners.	As dev cycle, + cve databases (NVD etc.)	N/A
7 Evolution	TBD	TBD	As dev cycle	Events from IoT Gateway sent to EI and in reverse direction.
8 Training & Education	N/A	N/A	Word, power point, internal streaming tooling	N/A

24.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 4

AHT-EPP	**SAP PM**	**Manual Work (MS Office)**	**Issue-Tracker (JIRA)**	**Revision Control System (git)**	**Build Server (jenkins)**	**Eclipse IOT Framework**	**Equipment Integration (EI)**	**Java Script, Java, Python**
1 Requirements		x	x					
2 Functional Design		x	x	x				
3 Procurement & Engineering	x	x	x	x	x	x		x
4 Deployment & Commissioning			x	x	x	x	x	x
5 Operation & Management		x	x	x	x	x		
6 Maintenance Decommissioning & Recycling	x	x	x	x	x	x		
7 Evolution		x	x	x	x	x	x	
8 Training & Education								

24.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 1

In the backend development, the development chain is well connected. Issues in the issue tracker allow to automatically create a new development branch in the revision control system, which automatically is built and tested by the build server. However, the link between requirements, architecture and deployment is currently not automated and needs to be tracked manually with word documents and automatic adjustments at the documents.

AHT-EPP	Gateway Software Components Toolchain	Hardware Components Toolchain	Backend Software Components Toolchain	Equipment Integration (EI) Toolchain
1 Requirements	Word: manual, Excel: manual	Word: manual, Excel: manual	Word: manual, Excel: manual, issue tracker: semi automatic, Modeling tool: semi automatic	Word: manual, Excel: manual, issue tracker: semi automatic
2 Functional Design	Word: manual, Excel: manual, issue tracker: semi automatic		Word: manual, Excel: manual, issue tracker: semi automatic, Modeling tool: semi automatic	Word: manual, Excel: manual
3 Procurement & Engineering	issue tracker: autmatic, git: autromatic with automated branching, buld server: automated build		issue tracker: autmatic, git: autromatic with automated branching, buld server: automated build, , Modeling tool:automatic code generation for parts	Word: manual, Excel: manual
4 Deployment & Commissioning	SSH, VPN: manual	dhcp: manual conf.	kubectl: semi-autmated deployment, modeling tool: semi automated configuration of backend system	SSH, VPN: manual
5 Operation & Management	monitoring: manual	monitoring: manual	prometheus: automated alerting, kubectl: semi-automated	TBD
6 Maintenance Decommissioning & Recycling	Maintenance for PoC done by engineering and external partners.		cve databases (NVD etc.): semi-automated notification	N/A
7 Evolution	TBD		N/A	N/A
8 Training & Education	N/A		Word, power point, internal streaming tooling: manual	N/A

24.4. Gap Analysis

24.4.1. Level of automation of the toolchain & Information passed

The traceability gap between the planning models and the actual implementation is a lack, which currently has to be filled manually. Apart from that is the deployment step currently a manual one, in which our software artifacts have to be distributed manually.

Requirements:
A requirements definition file in Excel is created which is passed to the next phase.
Functional Design:
The requirements in the Excel sheet are taken and transferred to a specification document, which details the requirements.
Procurement & Engineering:

The document is transferred to the procurement department, which uses the document to send out requests for quotations. Results are quotations, of which one is selected and an order is placed.
The result of the order is an implemented sub-component. However, this phase runs in cycles in which several words documents and emails are exchanged which detail the requirements and spec document.
Deployment & Commissioning:
The implemented sub-component is packaged as a tar.gz package, which is copied on a central FTP server, from which the component is copied to the actual system on which the component runs.
Operation & Management:
The running sub-component generates log files in text format and the running servers generate reports on health status. Both are in text format and written to the file system.
Maintenance:
If the subcomponent needs a bug fix an order is again placed and the process is rerun as described above.
Evolution:
Same here. If new features an order is placed.
Training & Education:
Training material for new sub-components are part of the initial order. It is mostly provided as word documents and accompanied by a powerpoint slide deck. For initial ramp up a training by the solution provider is done.

24.4.2. Level of integration with the Arrowhead Framework

As a baseline the use case does not use the Arrowhead framework. The realization with the Arrowhead framework will happen in the course of the project.

24.4.3. Missing Tools within the UC

Tooling to automate the deployment process.

Indispensable tools: Backend integration capabilities with data providing devices that allows to consume a huge amount of data and allows to scale out.

24.4.4. Missing Tools Outside the UC

A tool to allow for traceability from the requirements over the architecture to the actual running implementation.

24.5. Declared Tools

Name	Compatible with AHF	Inputs	Outputs
Current Sensor (Advantech WISE 4010)	No	Physical Quantity	Raw Measurement Data

Voltage Sensor (PicoScope)	No	Physical Quantity	Raw Measurement Data
IoT Edge Device (Kunbus RevPi)	No (Not Yet)	Raw Measurement Data from Sensors	Aggregated Data / MQTT / Events
Eclipse HONO	Yes	MQTT / JSON	AMQP messages for backend processing
Eclipse Ditto	Yes	AMQP messages	digital twin representation accessible via REST API
influxDB	No (Not Yet?)	AMQP messages	stored time series data
kubernetes	No (Not Yet?)	service description	running services

25. UC-16.2 [IFD]: IO link

Contact: Germar Schneider, Matthias Fehr, Dirk Mothes [germar.scheider@infineon.com]

The description of each Use Case can be found in D1.2. This is a new use case.

25.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 5

Block “Requirements”:
Collected specific details for the sensor use case. Which physical signal to measure? i.e. Voltage, Current, Pressure, Flow Temperature, Vibration a.s.o, Data Rates, Output Frequency, Where is the Sensor belonging to, Where to store the data, algorithms, purpose of the additional sensor signal.
Block “functional design”:
Match Sensor selection with available sensor hardware and interfaces/protocols. i.e. Hardware: µController Box or PLC (Siemens S71200, Beckhoff, ABB AC500) Interfaces/Protocols: Siemens API, ModbusTCP, OPC DA, ASCII data (General TCP)
Block “procurement & engineering”:
This Block includes the ordering of hardware as well as ordering all in house services from different departments (E.g. IT Software, Network Connections, Data key number configuration) Also it includes programming and configuring the sensor hardware or controllers. Sensor Installation and startup. Request all necessary software changes i.e. new protocols for EAF if the sensor could not be ordered with an existing one. Request LAN Port and IP Address or Name if DHCP is possible. Request for APC Suite (Advanced Process Control) configuration. Selecting Input streams and create calculated key numbers per stream i.e. Mean, min max, trend etc.., Installation of the sensor on the first tool, test dataflow and improve functionality as needed.
Block “deployment & commissioning”:
Ordering and installing hard and software for roll out the solution to multiple equipment/tools. Request for IT to rollout the tested setup to the desired equipment. Request APC Suite configuration for implementation of new sensor streams. Setup an operational concept.
Block “operation and management”:
Continuous data input is monitored by APC Checker software. Messaging per E-Mail and SMS if limit violations occur. Stability issues will be automatically highlighted by the system.
Block “evolution”:
Regularly review sensor data and limits my maintenance and process engineers. Improve key number calculation, limit setting and Run to Run Controllers. (R2R = Algorithms which control equipment parameter automatically (process recipes) in a given window based on sensor data
Block “training”:
Create training material in form of a presentation or a Wiki page. Inform relevant groups i.e. Maintenance and process engineers of the production department. Users that have to do with sensor integration, Databases or Equipment integration. Train 24*7 service. Train maintenance technicians in shift.

AHT-EPP	**Other Software** **(Putty, APC Tracker etc., Softing OPC Toolbox)**	**Manual Work**	**APC Trent Suite** **RunToRun Framework (R2R)**	**Sensor Software or Configuration Tools /IDE's** **(Arduino, ABB Automation Builder, DAVE4 / Eclipse, IFM LineRecorder Device)**	**Equipment Automation Framework (EAF)**
1 Requirements		#Collect required Sensor Signals # Understanding purpose of the sensor (goal) # Check ROI for Sensor Installation			
2 Functional Design	#Test Sensor Data Output (Putty, MQTT Broker etc.)	#Select Interfaces and Protocolls for POC # Create Service Order for new LAN Port and IP	#Request for new Sensor Integration #New key numbers	#programming µControllers or PLC if signal processing or conversion is necessary	Request for new data interfaces if not existing (TRAIL Request)
3 Procurement & Engineering		#Install Hardware, Set IP addresses,	#Configure data channels and key number calculations (manually by using UI)	#Configure sensor hardware i.e. IFM AL1342	#Build missing interfaces / functionalities as requeseted. Depends on priority of the project #Confirure data interface (Done by IT manually)
4 Deployment & Commissioning	Setup APC Tracker to monitor sensor data input continuously for data gaps.	Install additional hardware by "copy exactly"	#Configure limits and messaging for calculated key numbers	Copy Software to Sensor (Bootloader)	Subscribe to sensor data by selecting the topic (Site/Name of ProductionTool/
5 Operation & Management	Monitoring the Data availability, incidents, errors. (APC Checker)	Select data to be collected for training	# Automatic messaging on limit violation # Run to Run control. Change process recipe slightly on base of sensor data.		Data merging with tool datas and logistical metadata, Data storage into databases

6 Maintenance Decommissioning & Recycling		Maintenance Engineers are able to change hardware and do sensor configuration	24*7 Hotline available		24*7 Hotline
7 Evolution			# Review data (manually) #Improve Limit setting (manually) # Improve R2R algorithms	Upload AI Algorithm to Sensor / Edge Device if possible	
8 Training & Education		Setup training material and rules for Sensor integration, Create Wiki page and profide information to all relevant teams			

25.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 5

SAP is used for purchasing and order tracking as well as for warehouse management, parts tracking, maintenance action planning and tracking, MS Office tools is used in most cases for project planning and managing. But also share point lists are used. OPC DA is used for some sensor networks i.e. collection data from Siemens PLC or for IFM Vibration diagnostics (VSE002). All belongs for Equipment integration will be maintained in Visual Studio C#. If Input adaptors already exist, only the required class for this protocol needs to be selected and an XML file needs to be modified. If the protocol is not implemented a new sensor class needs to be written.

AHT-EPP	**SAP PM**	**Manual Work (MS Office)**	**IFM OPC Server, Softing OPC Server**	**Arduino, Infineon Dave4, ABB Automation Builder , IFM LR Device, Beckhoff TwinCat,**	**APC Suite and R2R Framework**	**C#**
1 Requirements		x				
2 Functional Design		x				
3 Procurement & Engineering	x	x	x	x	x	x
4 Deployment &			x	x	x	x

Commissioning						
5 Operation & Management			x	x	x	
6 Maintenance Decommissioning & Recycling	x	x			x	
7 Evolution		x			x	
8 Training & Education	x	x			x	

25.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 5

The tools used in the engineering phases “requirements”, “functional design”, “procurement & engineering” and “deployment & commissioning”, are not well connected as there exist no standard tools and experts do that in their own way, e.g. using programming tools they are familiar with. This also requires a lot of expert knowledge. The engineering phases “operation and management” and “ Maintenance Decommissioning & Recycling" are well connected in terms of preventive maintenance actions, tool down on limit violations, stability tracking (MTBF, Uptime etc.). Evolution will be done manually by reviewing sensor data by the process engineers. In Case of Run to Run Control the algorithm trains itself within an allowed process window. Training materials need to be set up manually but are booked automatically to the technicians. It is tracked that they complete the training within until a defined deadline. The YODA bus is used as a message service. It handles information like wafer and lot information, Workflow, recipes, Equipment status. It does not include sensor information. YODA (based on TIPCO) is mentioned here due to its importance for the Equipment and workflow automation. It’s the “backbone” of the automation. But for the engineering phases of the Arrrowhead use case it doesn’t play a rule. So that’s why it is not mentioned within the tables above about the engineering phases

AHT-EPP	**SAP**	**Manual Work**	**APC Trend Suite / Run2Run**	**YODA based on TIBCO**	**EAF (C#)**
1 Requirements					
2 Functional Design					
3 Procurement & Engineering	Ordering is partly automated	Sensor installation	Need to be configured manually		Setup need to be done manually

			for automated limit messaging and tool down booking		
4 Deployment & Commissioning	Reorder multiple times	Sensor installation/ Network configuration	Copy/ past		Rollout can be done by copy/paste
5 Operation & Management	Automatically maintenances actions, Automatic part ordering, automated tool status booking	No manual action	Automated limits, Automated process parameter adjustment (R2R), Connected to SAP Module	Service on TIBCO bus are providing available Wafers for the desired equipment. A frontend visualises and controls tool status like up, down, wait maintenance a.s.o.	Automatic mode for sensor reading and recording, database transfer etc. Connected to SAP module for Maintenance actions and to the TIBCO bus and its Frontend “Basiszelle“ for Tool down/up booking and Wafer delivery control
6 Maintenance Decommissioning & Recycling		Repair, preventive maintenance,		Equipment status (up, down , maintenance , engineering….)	
7 Evolution	automated reports (uptime, costs of ownership)	Review SAP and APC reports	Automated data reports	No action	Stability reports
8 Training & Education		Create training material for Maintenance, IT 24*7 Support, Sensor Configuration, Network architecture, Provide and commissioning of the material within the Infineon training centers.			

Toolchain	**SAP**	**Manual Work**	**APC Trend, R2R**	**YODA based on TIBCO (Multible Services)**	**EAF (VS C#)**
1		x			
2		x			
3	x	x	x		x

4	x		x		x
5	x		x	x	x
6	x	x	x	x	
7	x	x	x		
8		x			

25.4. Gap Analysis

25.4.1. Level of automation of the toolchain & Information passed

Each tool chain itself is highly integrated. (EAF, YODA Services, SAP) In terms of “operation and management” the tools are highly connected. The whole wafer production line is automated with these tools. Wafers / Lots are sent automatically to available and defined production tools. Production parameters are automated controlled. To achieve this high level of automation there are some barriers to take for a safe and stable sensor integration. So the sensor integration process is enforced with a lot of manual configuration and master data administration. The toolchain is also designed for providing a lot of information about the production process for tracing back process deviations and avoiding stability issues. So data reports are also available quickly and automated. AI is used more and more to automate even this process.
After collecting the requirements a sensor is selected. Existing solutions and protocols are screened. IP address, Protocol, Corresponding Equipment Name, X/Y Coordinates, Signal Names are passed over manually or per IT request form to engineering phases 3 and 4. Operation and management is fully automatic. E.g. equipment controlling, logging to databases, limit violations, in situ control of process parameters. So the output of EP 5 is the input of EP 5. It’s producing continuous information in this production phase. This information is reviewed in EP7 and given back as Input for EP5.

25.4.2. Level of integration with the Arrowhead Framework

None as of now for the facility use-case. For the PnP sensor integration use-case a complete implementation within AH FW is planned and currently developed as a proof-of-concept demonstrator application, including database services, protocol translating services, UUID Service (Hash ID) , merging and translation tools and visualization services

25.4.3. Missing Tools within the UC

Over 600 industry protocols are existing. Existing services for translating for Infineon relevant industry protocols to an MQTT SenML format are not known. If they do not exist they have to

be created. The most important protocols are Modbus TCP, OPC DA, OPC UA and IO_Link (IODD). So we need possibly Arrowhead Tools and services as follows - ModbusTCP --> MQTT (SenML), OPC DA --> MQTT (SenML), OPC UA --> MQTT(SenML).

The key services for the PnP sensor use case POC are: Translation services: ModbusTCP – MQTT (SenML), IO-Link – MQTT (SenML), Database Services MQTT(SenML) – APC File and GOETE Database

25.4.4. Missing Tools Outside the UC

For the initial POC we need Arrowhead Tools and services as follows - ModbusTCP --> MQTT (SenML) // OPC DA --> MQTT (SenML) // OPC UA --> MQTT(SenML) // IO-Link (MQTT) --> MQTT (SenML) // MQTT (SenML) to APC File. The APC File format can be read by the APC extractor. This tool is reading the file content and is calculating the run key numbers which were configured in the APC Trend Toolchain. So this service is important for getting the bridge from arrowhead to the productive sensor database. This service should be able to configure itself by the information provided by the sensor i.e. names in the field “n” are used as a header for the file column. The destination path of the file will be provided by the sensor. Possibly there is also a service MQTT(SenML) --> GOETE Database needed. This Database is an engineering database to evaluate AI algorithm// Service which possibly automatically identifies the protocol and links the matching translator. // Service which can filter for MQTT Topics by sending metadata like Tool Name or Building/ Floor etc.

25.5. Declared Tools

Name	Compatible with AHF	Inputs	Outputs
Modbus TCP Sensor	No	Binary Data	EP-O3(MQTT / SenML)
OPC UA PLC	No	OPC UA	EP-O3(MQTT /SenML)
IO-Link Sensors	No	MQTT / JSON	EP-O3(MQTT / SenML)
Sensor Querry	Yes	Search String / JSON with metadata (Toolname, Fab, Building ..)	Available MQTT Topics
Topic handle service	Yes	Topic	Topic as Hash ID / Handle

APC File Output	No	MQTT(SenML)	APC File Output. (ASCII File with header and footer)
Visualisation	Yes	DM-EP-O1	Graphics

26. UC-16.4 [HTW]: Alexa Voice Control of Chess Robot Yumi

Contact: Paul Patolla, Dirk Reichelt [Paul.patolla@htw-dresden.de, dirk.reichelt@htw-dresden.de]

The description of each Use Case can be found in D1.2. This is a new use case.

26.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 3

Block "Requirements":
Collected specific details for the use case. Evaluate existing demonstrator and software.
Block "functional design":
It should be only a demonstrator with no training or educational background.
Block "procurement & engineering":
Implementation of the required functions in the program. Outsourcing of the Chessboard recognition to Raspberry Pi.
Block "deployment & commissioning":
Roll out the program.
Block "evolution":
Bug fixing.

AHT-EPP	C#-Program	Manual Work	Chessboard recognition
1 Requirements	Voice Recognition with native APIs; Chess Engine; control the yumi robot; GUI to play chess; license-free; .NET framework 7.8		Recognize pieces on chessboard and provide piece matrix; accessible via API; runnable without GUI; Linux; recognition via OpenCV; license free; autorun
2 Functional Design	Monolithical		REST API to request piece matrix
3 Procurement & Engineering	Build a GUI monitor to monitor chess board/play chess; Manage yumi connections; display/say errors		#Configure data channels and key number calculations (manually by using UI)
4 Deployment & Commissioning	Installation von windows tablet		#Configure limits and messaging for calculated key numbers
5 Operation & Management	Monitor chess board; choose a yumi to play with; handle error on GUI; manage chess rules and moves; handle voice input/output	Set chess pieces on the board; Start C#-Program and choose Yumi interface from list	Response a chess piece matrix after each move
6 Maintenance Decommissioning & Recycling	Bugfixing		Bugfixing

7 Evolution			
8 Training & Education			

26.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 3

Only the ABB robot studio needs to be adopted. This tool supports the simulation of an ABB Yumi.

AHT-EPP	**ABB Robotstudio**
1 Requirements	x
2 Functional Design	x
3 Procurement & Engineering	x
4 Deployment & Commissioning	x
5 Operation & Management	
6 Maintenance Decommissioning & Recycling	
7 Evolution	x
8 Training & Education	

26.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 3

The ABB Robotstudio is used in AHT-EPP 1, 2, 3, 4 and 7. It is only used to develop new functions and designs and is not part of the production environment. So there is no automation needed.

AHT-EPP	**ABB Robotstudio**

1 Requirements	Simulate whole functionality of Yumi; accessible from external programs
2 Functional Design	
3 Procurement & Engineering	order required licenses; implement necessary functions
4 Deployment & Commissioning	Installation on windows system; create yumi environment/system
5 Operation & Management	
6 Maintenance Decommissioning & Recycling	
7 Evolution	Used to develop new functions outside of the production environment
8 Training & Education	

26.4. Gap Analysis

26.4.1. Level of automation of the toolchain & Information passed

The automation and integration level of the first toolchain is very high. Information passed automatically from system to system. The user has only to choose the yumi robot interface and set the initial state of the physical chessboard. The ABB Robotstudio as a second toolchain is not automated nor highly integrated. It supports the developer to implement and test new functions or searching for bugs in the robot control source code. Changes will manually be implemented in the C#-Program.

There is no automatic monitoring or transmission of information on the individual phases. Ideas and improvements have to be brought to the developer, which he then implements. The information is managed via a manual processing list.

26.4.2. Level of integration with the Arrowhead Framework

The Use Case should be completely integrated. Non-integrable systems are integrated with the help of a wrapper. Further systems are not planned for integration for the time being.

26.4.3. Missing Tools within the UC

N/A

26.4.4. Missing Tools Outside the UC

Use Case driven Web Interface for Core Services, NTP Core Service, C# AHT Client/Library

26.5. Declared Tools

Name	Compatible with AHF	Inputs	Outputs
Chess Board Matrix	yes		EP-O3 (JSON)
Speech Data	yes		EP-O3 (SSML)
Piece Move	yes		EP-O3 (JSON)
Yumi commands	no	EP-O2	EP-O3

26.6.

27. UC-16.5 [TUD]: Automated Material Handling System

Contact:

The description of each Use Case can be found in D1.2. This is a new use case.

27.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 2

Block “Requirements”:
Relevant aspects for predictive maintenance are machine type, kinematic properties and production metadata. Further typical damages and faults of the machines under consideration from historical experience are helpful.
Block “functional design”:
This block is less relevant in this use case, it can be used to test and evaluate additional data analysis methods.
Block “procurement & engineering”:
This step includes the selection of diagram types based on the concrete machine including properties as rotation frequency and external noise sources. Additionally, feature evaluation methods have to be selected, i.e. KPIs that are relevant for condition estimation of the machine type.
Block “deployment & commissioning”:
The hypothesis consolidation rules have to be adapted to the use case, i.e. the rules of pattern recognition for creating final hypotheses from diagram-specific primary hypotheses. In addition to that, the hypothesis generators have to be parameterized.
Block “operation and management”:
Based on breakdowns typical vibration patterns preceding the breakdown can be learned for improving the prediction quality of the wear model.
Block “maintenance”:
The algorithms and platform have regularly been brought to the latest revision. The vibration sensors have to be checked regularly. After changes of the machine, changes of the model properties may be necessary.
Block “evolution”:
Operation and management usually leads to new information about wear relevant aspects of the machine. This information can be used to improve the analysis software.
Block “training”:
Different stakeholders have to cope with the software, reaching from operation personal overdiagnosis experts to managers. This requires training material for each kind of user

AHT-EPP	PDM / condition monitoring in Semicond. Facility	Measurement FOUP Integration / Vibra. Analysis in AMHS
1 Requirements	Collect machine type, kinematic properties and production meta data	Requirements are collected regarding measurement frequency, necessary data to merge, desired visualization

2 Functional Design	Matlab will be used to test and evaluate new algorithms	Design of the toolchain for automated vibration measurement on simulated environments (SBCs)
3 Procurement & Engineering	selection of hardware and diagnostic diagram types based on the concrete machine	Missing features of the measurement toolchain have to be retrofitted, e.g., WLAN functionality for the meas. FOUP
4 Deployment & Commissioning	Parameterization and adaption of diagnosis hypothesis generator	The toolchain will be transferred from PoC phase to the server environment of the semiconductor manufacturing plant and tested in different conditions, e.g., in areas sparsely covered with WLAN
5 Operation & Management	improving the prediction quality by observing typ. Vibration patterns	Will be performed by maintenance engineers and AMHS experts of the fab
6 Maintenance Decommissioning & Recycling	Check vib. Sensors regularly, adopt model properties to changed operating conditions	Will be performed by maintenance engineers and AMHS experts of the fab
7 Evolution	Incorporate new information relevant to wear into analysis algorithms	Additional requirements or inadequacy of the toolchain will be collected and subsequently remedied
8 Training & Education	Provide for different stakeholders, e.g., operation and maintenance personnel	Trainings and training material on the usage of the AMHS vibration monitoring tool chain will be provided

27.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 2

The provided solution will be based on the Conimon WebApp. In addition, Matlab is used as a reference as Matlab allows the simple test of new algorithms before integrating them into the more complex WebApp infrastructure.

AHT-EPP	**PDM / condition monitoring in Semicond. Facility**	**Measurement FOUP Integration / Vibra. Analysis in AMHS**
1 Requirements	MS Word, Excel, Etherpad and Adobe PDF are used to collect the necessary data sheets and requirements	MS Word, Excel, Etherpad and Adobe PDF are used to collect the necessary data sheets and requirements
2 Functional Design	Matlab will be used to test and evaluate new algorithms	Eclipse IDE, Hibernate, GIT and diverse Programming tools are used to implement the PoC

3 Procurement & Engineering	Matlab and the Conimon Webapp to select and instrument the required analysis algorithms	Eclipse IDE, Hibernate, GIT and diverse Programming tools are used to implement the remaining features
4 Deployment & Commissioning	Parameterization and adaption of diagnosis hypothesis generator within the Conimon Webapp	Various tools of the manufacturing plants IT landscape as well as the AH Framework to roll out the measurement toolchain
5 Operation & Management	improving the prediction quality by observing typ. Vibration patterns within Matlab and the Conimon Webapp	Various tools of the manufacturing plants IT landscape for operation. Ms Word and Etherpad to document problems and enhancement suggestions
6 Maintenance Decommissioning & Recycling	Monitoring done with the Conimon Webapp	See above
7 Evolution	Eclipse IDE, Hibernate, GIT and diverse Programming tools to implement enhancements	Eclipse IDE, Hibernate, GIT and diverse Programming tools to implement enhancements
8 Training & Education	MS Word, Etherpad and Adobe PDF are used to create and distribute documentation, Video editing and Zoom for Webinars and educational videos	MS Word, Etherpad and Adobe PDF are used to create and distribute documentation, Video editing and Zoom for Webinars and educational videos

27.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 2

The tools used in the engineering phases “requirements”, “functional design”, and “procurement & engineering” are not well connected as there exist no standard tools and experts do that in their own way, e.g. using Excel tables, Word documents or Matlab scripts. This also requires a lot of expert knowledge. The engineering phases “deployment & commissioning”, and “operation and management” are well connected as they are done usually with the same software. “Evolution” and “training” are not supported by use case specific tools and thus independent from the tools of the other engineering phases.

AHT-EPP	**PDM / condition monitoring in Semicond. Facility**	**Measurement FOUP Integration / Vibra. Analysis in AMHS**
1 Requirements	Manual usage	Manual usage
2 Functional Design	Manual usage of Matlab	Git various other libraries and tools are integrated through the Eclipse IDE, i.e. automatically connected

3 Procurement & Engineering	Manual connection of Matlab and the Conimon Webapp	Git various other libraries and tools are integrated through the Eclipse IDE, i.e. automatically connected
4 Deployment & Commissioning	Conimon Webapp supports semi-automatic configuration depending on measurement hardware	Various tools of the manufacturing plants IT landscape and the measurement FOUP shall be interconnected via the AH Framework
5 Operation & Management	Conimon Webapp supports semi-automatic configuration depending on measurement hardware	Various tools of the manufacturing plants IT landscape and the measurement FOUP shall be interconnected via the AH Framework
6 Maintenance Decommissioning & Recycling	Conimon Webapp supports semi-automatic configuration depending on measurement hardware	See above
7 Evolution	Git various other libraries and tools are integrated through the Eclipse IDE, i.e. automatically connected	Git various other libraries and tools are integrated through the Eclipse IDE, i.e. automatically connected
8 Training & Education	Manual usage of the tools	Manual usage of the tools

27.4. Gap Analysis

27.4.1. Level of automation of the toolchain & Information passed

For the use-case regarding the vibration measurement of facility equipment, there are a number of different proprietary solutions that support several steps of the toolchain. The integration level inside each of these tools is high. However, the tools are not compatible with each other and only raw data can be exchanged in a more or less practical form. The late phases like pattern recognition and training are not supported by most of the tools. For the monitoring of vibrations within the AMHS, the current, manual process shall be fully automated with the help of the AH framework. At the moment, only the subsequent data processing steps, starting from the acquisition of vehicle movement data until data visualization, are somewhat automated in a proprietary way within the factories IT landscape. The missing steps involve the automated acquisition of vibration measurement data from the instrumented FOUP and the aggregation of this data with necessary meta information. This will be done by using AH Framework services and is currently in the PoC implementation phase.

From the modeling phase, meta data about the machine, e.g. type and kinematics, is passed to the subsequent phases. After data acquisition, the measurements are added. After diagram preparation, the diagrams or at least the relevant parts of them are passed to the evaluation steps. Finally, primary hypotheses created in the feature extraction phase are used in the pattern recognition phase. As the FOUP vibration measurement use-case is comparatively

small and isolated, the information about requirements, feedback for further development and the rollout of software versions is done manually in a relatively small team

27.4.2. Level of integration with the Arrowhead Framework

None as of yet for the facility use-case, experimentation to automate certain aspects of this rather complex workflow will be done later in year 2 of the project. For the FOUP vibration measurement use-case in the material handling system, a complete implementation within AH FW is planned and currently developed as a proof-of-concept demonstrator application, including database services, real time services, data generators, merge tools and visualization services

27.4.3. Missing Tools within the UC

An improved automation in the later phases 4-7 is realistic. Up to now, no software is able to create hypotheses about damages and faults in machines used in semiconductor production. This requires the implementation or selection and parametrization of algorithms suited for that use case. This is the goal of TUD in the Arrowhead tools project. In the Conimon WebApp first steps in the desired direction are available, i.e. a collection of algorithms and the methodology for formulating pattern recognition rules in form of XML sheets. For the FOUP vibration measurement use-case, AH Framework services such as a streaming Database for transient vibration measurement data (time series) and clock synchronization services are required. These are currently implemented as a PoC by members of the use-case themselves

The works will be based on the Conimon WebApp as this tool is closest to the goal of TUD in Arrowhead tools and provides the flexibility to extend it for the purposes of the new use case. It is also the only one providing the necessary support for diagnosis hypothesis of higher orders and the required rule description language for advanced machine diagnosis

27.4.4. Missing Tools Outside the UC

The automation of the early phases 1-3 (like modeling and parametrization) as well as 8 would be desirable but this is extremely case specific such that an automation in near future cannot be expected. A (graphical) tool for an easier retrieval, configuration and “connection” of services would be appreciated,

27.5. Declared Tools

Since the use-cases of monitoring of machine health status (AMHS and facility) are quite singular as a whole, the developed services as planned of now are described instead, since they promise a certain re-usability within the automation of future AHT-Eps (External: Provided as Data sources or –Sinks by the manufacturing plant, Tools serve as gateways or SW-adapters)

Name	This is a tool in the following	Compatible with AHF	Inputs	Outputs

	phase:			
Measurement-FOUP automation	StkH3-EPP 1, 2, 3, 4	Yes	FOUP Measurement Control (external)	MF-EP-O1(Vibration time series, CSV data)
Vehicle Positions provider service	StkH3-EPP 2, 3, 4	Yes	External	VP-EP-O1(Time – Vehicle – Position tuple, CSV data)
AMHS Layout provider service	StkH3-EPP 2, 3, 4	Yes	External	AL-EP-O1 (Coordinates as Geolocations, CSV)
Streaming DB – Timeseries service	StkH3-EPP 2, 3, 4	Yes	MF-EP-O1	DB-EP-O1
Data Merging Algorithm and service	StkH3-EPP 2, 3, 4	Yes	DB-EP-O1; VP-EP-O1; AL-EP-O1	DM-EP-O1
Adapter service for On-Site Visualisation tool	StkH3-EPP 2, 3, 4	Yes	DM-EP-O1	Graphics, External

28. UC-16.6 [UzL]: Tester

Contact: Mladen Berekovic, Javad Ghofrani [berekovic@iti.uni-luebeck.de, ghofrani@iti.uni-luebeck.de]

The description of each Use Case can be found in D1.2. This is a new use case which is still in the conceptual phase.

28.1. Gap Analysis

28.1.1. Level of automation of the toolchain & Information passed

Getting an overall overview of the system and extracting the functionalities is not a simple task which can be automated. There is a high need of experience in architecting distributed systems and system engineering. Separation of the monolithic system into some isolated services is the most difficult part of which can not be automated with tools

The information is passed through emails, meetings and protocols. For managing the meetings we used webex and captured the meetings with OBS studio. Microsoft word and Microsoft excel is used to capture the information as meeting protocols

28.1.2. Level of integration with the Arrowhead Framework

There is not yet any usage of arrowhead framework. For handling the service discovery and broadcasting the sensor and machine data, captured from PLCs and monitoring systems, the AHF will be used.

28.1.3. Missing Tools within the UC

(1) Converting the system interfaces into proper interfaces which are usable by arrowhead framework
(2) Automation of the onboarding process for service consumer in arrowhead framework

28.1.4. Missing Tools Outside the UC

Assistant system/guidelines or tools which support the system engineers with taking decisions about dividing system architectures into services / smaller pieces /separated functionalities / Microservices

28.2. Declared Tools

To be defined.

29. UC-16.7 [EDMS]: DTX and MAGICFLOW

Contact: Vahid Salehi [v.salehi@edmrc.de]

The description of each Use Case can be found in D1.2. This is a new use case.

29.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 5

Block “Requirements” with MAGICFLOW:

Collected specific details for the sensor use case. Which physical signal to measure? i.e. Voltage, Current, Pressure, Flow Temperature, Vibration a.s.o, Data Rates, Output Frequency, Where is the Sensor belonging to, Where to store the data, algorithms, purpose of the additional sensor signal.

Block “functional design” MAGICFLOW:

Match Sensor selection with available sensor hardware and interfaces/protocols. i.e. Hardware: DTX-Box or PLC MAGICFLOW

Block “procurement & engineering”:

This Block includes the ordering of hardware as well as ordering all in house services from different departments (E.g. IT Software, Network Connections, Data key number configuration) Also it includes programming and configuring the sensor hardware or controllers. Sensor Installation and startup. Request all necessary software changes i.e. new protocols for DTX if the sensor could not be ordered with an existing one. Request LAN Port and IP Address or Name if DHCP is possible.

Block “deployment & commissioning” MAGICFLOW:

Ordering and installing hard and software for roll out the solution to multiple equipment/tools. Request for IT to rollout the tested setup to the desired equipment.

Block “evolution”:

Regularly review sensor data and limits my maintenance and process engineers. Improve key number calculation, limit setting and Run to Run Controllers.

Block “training”:

Create training material in form of a presentation or a Wiki page. Inform relevant groups i.e. Maintenance and process engineers of the production department. Users that have to do with sensor integration, Databases or Equipment integration. Train maintenance technicians in shift.

AHT-EPP	Other Software (Putty, Sick, TOF)	Manual Work	APC Trent Suite RunToRun Framework (R2R)	Sensor Software or Configuration Tools /IDE’s (Arduino, ABB Automation Builder, DAVE4 / Eclipse, IFM	Equipment Automation Framework (EAF)

				LineRecorder Device)	
1 Requirements		#Collect required Sensor Signals # Understanding purpose of the sensor (goal) # Check ROI for Sensor Installation			
2 Functional Design	#Test Sensor Data Output (Putty etc.)	#Select Interfaces and Protocolls for POC # Create Service Order for new LAN Port and IP	#Request for new Sensor Integration #New key numbers	#programming µControllers or PLC if signal processing or conversion is necessary	Request for new data interfaces if not existing (TRAIL Request)
3 Procurement & Engineering		#Install Hardware, Set IP addresses,	#Configure data channels and key number calculations (manually by using UI)	#Configure sensor hardware i.e. IFM	#Build missing interfaces/functionalities as requested. Depends on priority of the project #Confirure data interface (Done by IT manually)
4 Deployment & Commissioning	Setup APC Tracker to monitor sensor data input continuously for data gaps.	Install additional hardware by “copy exactly”	#Configure limits and messaging for calculated key numbers	Copy Software to Sensor (Bootloader)	Subscribe to sensor data by selecting the topic (Site/Name of ProductionTool/
5 Operation & Management	Monitoring the Data availability, incidents, errors.	Select data to be collected for training	# Automatic messaging on limit violation # Run to Run control. Change process recipe slightly on base of sensor data.		Data merging with tool data and logistical metadata, Data storage into databases
6 Maintenance Decommissioning & Recycling		Maintenance Engineers are able to change hardware and do sensor configuration	Hotline available		Hotline
7 Evolution			# Review data (manually) #Improve Limit setting (manually) # Improve R2R algorithms	Upload AI Algorithm to Sensor / Edge Device if possible	

8 Training & Education		Setup training material and rules for Sensor integration, Create Wiki page and provide information to all relevant teams			

29.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 5

DTX Platform is used for visualization of the sensor data, MS Office tools is used in most cases for project planning and managing. But also share point lists are used. OPC DA is used for some sensor networks i.e. collection data

AHT-EPP	**DTX Platform**	**MAGICFLOW Tool**	**Commercial Cloud server 1u1**	**Raspberry Pie, 3D developed sensors**	**Sick company**	**Json, Python**
1 Requirements	x	x	x			
2 Functional Design	x	x	x			
3 Procurement & Engineering	x	x	x		x	x
4 Deployment & Commissioning		x	x	x	x	x
5 Operation & Management		x	x	x	x	x
6 Maintenance Decommissioning & Recycling	x	x			x	
7 Evolution		x	x		x	
8 Training & Education	x	x	x		x	

29.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 3

The tools used in the engineering phases “requirements”, “functional design”, “procurement & engineering” and “deployment & commissioning” are based on Magic flow which is from AI Digital Solutions developed by Prof. Dr. Vahid Salehi. With this approach it is possible to develop all the phases of the AH Tools system development.

AHT-EPP	DTX Platform	Manual Work	MAGICFLOW
1 Requirements			
2 Functional Design			
3 Procurement & Engineering	Visualization of the sensor	Sensor installation	Need to be configured manually for automated limit message
4 Deployment & Commissioning	Programming the interface	Sensor installation/ Network configuration	Copy/ past
5 Operation & Management	Automatically maintenances actions, Automatic part ordering, automated tool status booking	No manual action	Req, Function and Arch definition.
6 Maintenance Decommissioning & Recycling		Repair, preventive maintenance,	
7 Evolution	automated reports (uptime, costs of ownership)	Review SAP and APC reports	Automated data reports
8 Training & Education		.	

29.4. Gap Analysis

29.4.1. Level of automation of the toolchain & Information passed

Each tool chain itself is highly integrated. (DTX Platform, MAGICFLOW) In terms of “operation and management” the tools are highly connected. The whole wafer production line is automated with these tools.

After collecting the requirements a sensor is selected. Existing solutions and protocols are screened. IP address, Protocol, Corresponding Equipment Name, Signal Names are passed over manually or per IT request.

29.4.2. Level of integration with the Arrowhead Framework

None as of yet for the facility use-case. For the DTX Platform and sensor integration use-case a complete implementation within AH FW is planned and currently developed as a proof-of-concept demonstrator application, including database services, protocol translating services, UUID Service, merging and translation tools and visualization services

29.4.3. Missing Tools within the UC

Over 600 industry protocols are existing. Existing services for translating for Infineon relevant industry protocols to a SenML format are not known. If they do not exist they have to be created. The most important protocols are Modbus TCP, OPC DA, OPC UA and IO_Link (IODD).

29.4.4. Missing Tools Outside the UC

For the initial POC we need Arrowhead Tools and services as follows - Modbus TCP --> DTX Platform // OPC DA --> DTX Platform // OPC UA --> DTX Platform // MAGICFLOW,The APC File format can be read by the APC extractor. This tool is reading the file content and is calculating the run key numbers which were configured in the APC Trend Toolchain.

29.5. Declared Tools

For the monitoring of machine health status (AMHS and facility), no tool development in the AH Tools sense is planned, the developed services are described instead

Name	Compatible with AHF	Inputs	Outputs
Modbus TCP Sensor	No	Binary Data	EP-O3
OPC UA PLC	No	OPC UA	EP-O3
IO-Link Sensors	No	JSON	EP-O3
Sensor Querry	Yes	Search String / JSON with metadata (Toolname, Fab, Building ..)	--
Topic handle service	Yes	Topic	Topic as Hash ID / Handle
APC File Output	No	DTX Platform	APC File Output. (ASCII File with header and footer)

Visualisation	Yes	DM-EP-O1	Graphics

30. UC-16.X [IFAK]: Legacy System Integration With Extended Historian Service

Contact: Mario Thron, Christian Hübner, Thomas Bangemann [Mario.Thon@ifak.eu]

The description of each Use Case can be found in D1.2. This is a new use case.

30.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 5

AHT-EPP	Extended Historian Service (EHS)	EHS Configuration system (EHS-CS)	Production System	Data Analysis Tools	EHS Adapters (source and sink adapters)
1 Requirements	Requirements are collected in WP9 and WP3 of the AHT project.	Requirements are collected in WP9 and WP5 of the AHT project.	StkH1 defines the need for optimization of the production system in order to reach goals regarding several optimization criteria like minimization of material, energy and the like or performance enhancements.		Requirements are collected in WP9 and WP3 of the AHT project.
2 Functional Design	The functional design is done in WP3 of AHT. There is a specification document.	The functional design is done in WP5 of AHT. A specification document is work in progress.	StkH3 defines needed access to Arrowhead application services, fieldbus systems and control systems.	StkH2 analyses the production system structure and specifies influencing parameters stemming from the automation systems. He additionally designs the necessary algorithms as mathematical expressions.	The functional design is done in WP3 of AHT. There is a specification document.
3 Procurement & Engineering	The procurement and engineering is done in AHT WP3.	The procurement and engineering is done in AHT WP5.	StkH3 changes programs in a manner that cyclically generated information from the data analysis tool (e.g. predictions) will lead to an optimized production process. Since the creation of optimization information will be less reliable, there will be default values in case of absence of the optimization information.	StkH2 develops the algorithms defined in AHT-EPP 2 as source code for the data analysis tool.	StkH3 exports variable lists out of the communication or control configurations. This may be a network configuration file or a PLC variable or I/O list. Those lists are transformed by StkH3 into adapter configuration files.
4 Deployment & Commissioning	StkH2 or StkH3 deploy the EHS together with the Arrowhead Framework on a computing resource within the Intranet of the producing company.	StkH2 or StkH3 deploy the EHS-CS on any computing resource within the Intranet of the producing company.	StkH3 installs the changed programs into the control systems.	StkH2 deploys the data analysis software on any computing resource within the Intranet of the producing company. The algorithm source codes are put on the right places.	StkH3 deploys the source adapters on any computing resource within the Intranet of the producing company. Then the configuration files are put in the right place. StkH2 installs the right sink adapter in the data analysis tool.

5 Operation & Management	StkH2 or StkH3 start the EHS together with the Arrowhead Framework.	StkH2 or StkH3 start the EHS-CS.	StkH1 changes the operation procedures based on the information got by StkH2. Cyclically generated optimization information will influence the control systems automatically.	StkH2 produces the information on how to change the operation procedures in order to optimize the production process or he starts the cycle of information generation.	StkH3 starts the source adapters. Sink adapters are started in scope of the data analysis software.
6 Maintenance Decommissioning & Recycling		StkH2 or StkH3 can use the EHS-CS to monitor problems in the EHS or the adapters.		StkH2 uses the debugging facilities of the data analysis software to repair failures.	
7 Evolution	Feature requests are handled over the software development resource (probably GitHub).	Feature requests are handled over the software development resource (probably GitHub).	Evolution starts by changed requirements, so StkH1 will define new requirements and initiate further development of all necessary components (EHS and adapters).	Evolution of data analysis software is out of scope of AHT.	Feature requests are handled over the software development resource (probably GitHub).
8 Training & Education	The software will be provided including installation and usage manuals. The code will be in-line documented. Missing information can be requested as bug-report in the software development resource (probably GitHub).	The software will be provided including installation and usage manuals. The code will be in-line documented. Missing information can be requested as bug-report in the software development resource (probably GitHub).		Training regarding data analysis software is out of scope of AHT. There are typically various videos available on-line via video-sharing platforms like YouTube.	The software will be provided including installation and usage manuals. The code will be in-line documented. Missing information can be requested as bug-report in the software development resource (probably GitHub).

30.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 5

AHT-EPP	Extended Historian Service (EHS)	EHS Configuration system (EHS-CS)	Production System	Data Analysis Tools	EHS Adapters (source and sink adapters)
1 Requirements	**Microsoft Word** is used for the requirements specification.	**Microsoft Word** is used for the requirements specification.	**Microsoft Word** is used for the requirements specification.		**Microsoft Word** is used for the requirements specification.
2 Functional Design	**Microsoft Word** and **Powerpoint** are used for description of the functional design.	**Microsoft Word** and **Powerpoint** are used for description of the functional design.	**Microsoft Word** and **Powerpoint** are used for description of the functional design.	**Microsoft Word** and **Powerpoint** are used for description of the functional design.	**Microsoft Word** and **Powerpoint** are used for description of the functional design.

3 Procurement & Engineering	**Visual Studio Code** is used as IDE for software development (Java). **Maven** is used as build tool and package management tool. **Git** is used as version control system.	**Visual Studio Code** is used as IDE for software development (Java). **Maven** is used as build tool and package management tool. **Git** is used as version control system.		**Visual Studio Code** is used as IDE for software development (Python). Alternatively **Jupyter Notebook** is used for code development. **Git** is used as version control system.	**Visual Studio Code** is used as IDE for software development (Java). **Maven** is used as build tool and package management tool. **Git** is used as version control system.
4 Deployment & Commissioning	**OpenJDK** is used as run-time system.	**OpenJDK** is used as run-time system.	The demonstrator will use **Siemens PLCs** as runtime system for automation control programs and **Python 3** as run-time systems for mock-up sensors.	A current version of **Python 3** is used as run-time system. An optional alternative is to use **GNU R** as run-time system.	**OpenJDK** is used as run-time system.
5 Operation & Management	**Shell scripts** will be used to start the system on the run-time system used for deployment.	**Shell scripts** will be used to start the system on the run-time system used for deployment.	The **Siemens PLCs** will start running by switching the power-button on. The mockup-sensors will be started by **Shell scripts** started over an **SSH** access provided by the underlying operating systems.	The **Python 3** scripts are directly interpreted by the run-time system. The starting is triggered by StkH2.	**Shell scripts** will be used to start the system on the run-time system used for deployment.
6 Maintenance Decommissioning & Recycling	**E-Mails** to the developers will be used to reports to the developers in early stages of the project. Later on **GitHub** bug or feature item handling will be used for that purpose.	**E-Mails** to the developers will be used to reports to the developers in early stages of the project. Later on **GitHub** bug or feature item handling will be used for that purpose.	**E-Mails** to the developers will be used for reporting of bugs.	**E-Mails** to the developers will be used for reporting of bugs.	**E-Mails** to the developers will be used to reports to the developers in early stages of the project. Later on **GitHub** bug or feature item handling will be used for that purpose.
7 Evolution	**GitHub** feature item handling will be used for reporting feature wishes.	**GitHub** feature item handling will be used for reporting feature wishes.	**E-Mails** to the developers will be used for reporting of feature wishes.	**E-Mails** to the developers will be used for reporting of feature wishes.	**GitHub** feature item handling will be used for reporting feature wishes.
8 Training & Education	**Visual Studio Code** will be used to write installation and usage manuals based on the **Markdown** and/or **Emacs Org-Mode** formats. Those formats are later transformed in a visual appealing format like **HTML**.	**Visual Studio Code** will be used to write installation and usage manuals based on the **Markdown** and/or **Emacs Org-Mode** formats. Those formats are later transformed in a visual appealing format like **HTML**.		**Jupyter Notebook**s are used for coding and training.	**Visual Studio Code** will be used to write installation and usage manuals based on the **Markdown** and/or **Emacs Org-Mode** formats. Those formats are later transformed in a visual appealing format like **HTML**.

30.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 5

AHT-EPP	Extended Historian Service (EHS)	EHS Configuration system (EHS-CS)	Production System	Data Analysis Tools	EHS Adapters (source and sink adapters)
1 Requirements	**Manual work.**	**Manual work.**	**Manual work.**		**Manual work.**
2 Functional Design	**Manual work.**	**Manual work.**	**Manual work.**	**Manual work.**	**Manual work.**
3 Procurement & Engineering	Coding: **manual** work Solving library dependencies: **automatic** (Maven) Building: **automatic** (Maven)	Coding: **manual** work Solving library dependencies: **automatic** (Maven) Building: **automatic** (Maven)		Coding: **manual** work Solving library dependencies: **automatic** (pip3). Building: **not necessary** (interpreted language).	Coding: **manual** work Solving library dependencies: **automatic** (Maven) Building: **automatic** (Maven)
4 Deployment & Commissioning	Deployment: **automatic**.	Deployment: **automatic**.	Deployment: **automatic**.	Deployment: **automatic**.	Deployment: **automatic**.
5 Operation & Management	**Start-up: automatic via shell script.**	**Start-up: automatic via shell script.**	**Start-up: automatic via power-on button.**	**Start-up: automatic by Jupyter Notebook.**	**Start-up: automatic via shell script.**
6 Maintenance Decommissioning & Recycling	**Manual work.**	**Manual work.**	**Manual work.**	**Manual work.**	**Manual work.**
7 Evolution	Iterative approach starting with a change request (**manual**) then according to steps 1 to 5 of this description.	Iterative approach starting with a change request (**manual**) then according to steps 1 to 5 of this description.	Iterative approach starting with a change request (**manual**) then according to steps 1 to 5 of this description.	Iterative approach starting with a change request (**manual**) then according to steps 1 to 5 of this description.	Iterative approach starting with a change request (**manual**) then according to steps 1 to 5 of this description.
8 Training & Education	**Automatic** transformation of **manually** created files.	**Automatic** transformation of **manually** created files.	**Automatic** transformation of **manually** created files.	**Automatic** transformation of **manually** created files.	**Automatic** transformation of **manually** created files.

30.4. Gap Analysis

30.4.1. Level of automation of the toolchain & Information passed

Coding will still be manual work. All other processes are automated by using the IDE or shell scripts.

Specifications are written manually and checked into the source code repositories. Bug- and feature requests are managed via e-mail in the first project stage, later on in via bug and feature item management of the version management system (GitHub). Source code is created manually and deployed automatically by use of IDE features. System start-up is realized by start-up scripts.

30.4.2. Level of integration with the Arrowhead Framework

The toolchain integrates with the Arrowhead Framework data producers by use of source adapters. Information can be fed back into the Arrowhead Framework by development of sink adapters. Source adapters are integral part of the EHS development process within the AHT project. Also sink adapters will be developed, but for access by data analytics software. Sink adapters, which provide information back into the Arrowhead Framework are currently out of scope due to the necessary efforts, but the basic building blocks are created.

30.4.3. Missing Tools within the UC

The Arrowhead Framework core services (especially service registry, authorization and orchestration systems) are critical for our use case. Changes in their interfaces would affect our developments.

30.4.4. Missing Tools Outside the UC

N/A

30.5. Declared Tools

Name of the tool	This is a tool in the following phase:	Compatible with AHF	Inputs	Outputs
Extended Historian Service (EHS)	Procurement and engineering	Yes	None	Time series of sensor values, published via an EHS specific interface.
EHS Configuration system (EHS-CS)	Functional design and procurement and engineering	Yes	None	Configuration script for a specific EHS set-up.
Production System	Requirements	No, considered as system to be adapted by EHS to get AHF compatibility.	None	Sensor signals
Data Analysis Tools	Requirements	Not intended, data analytics tool will be adapted via EHS to the AHF.	Time series of sensor values provided by the EHS over EHS sink adapters in a specific format, which can be interpreted by the data analysis tool.	Reports about the time series indicating, which part of the production process can be optimized by influencing which process parameter.

EHS Adapters (source and sink adapters)	Procurement and engineering	Yes	Source adapters: sensor values transmitted over a legacy communication system Sink adapters: time series values in an EHS specific encoding	Source adapters: sensor values transmitted in a format, which is accepted by the EHS Sink adapters: time series values in a format, accepted by the data analysis tool

31. UC-17 [IFAT, AEE, EQUA, EQCH, AIT, FB]: Linking a Building Simulator to a Physical Building in Real-Time

Contact: Dagmar Jaehnig [d.jaehnig@aee.at]

The description of each Use Case can be found in D1.2

31.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 1

AHT-EPP	Optimization of Building Performance
1 Requirements	StkH4 acquires documentation of building and building services and discusses with StH3 the desired room conditions.
2 Functional Design	StkH54develops a plan for the monitoring campaign including number and type of sensors needed, data logging system, Data transfer etc. (in cooperation with StkH3)
3 Procurement & Engineering	StkH4 selects and orders the hardware and software components necessary for the monitoring campaign.
4 Deployment & Commissioning	Installation of the hard and software components by StkH4 (in cooperation with StkH3)
5 Operation & Management	StkH4 surveys data acquisition, analyzes the data and proposes optimization of control strategies, setpoints or even necessary changes in the building services.
6 Maintenance Decommissioning & Recycling	Deinstallation of monitoring equipment by StkH4, equipment can be reused for other projects
7 Evolution	--
8 Training & Education	--

31.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 1

AHT-EPP	Optimization of Building Performance toolchain
1 Requirements	**CAD software**: Drawings of building and HVAC equipment **Microsoft Word** or **Acrobat Reader**: Documentation of implemented control strategy
2 Functional Design	**Microsoft Word** and **Excel:** Documentation of monitoring equipment needed, list of sensors

	etc.
3 Procurement & Engineering	none
4 Deployment & Commissioning	Software for programming of **data logging system**
5 Operation & Management	**Data analysis tools, database tools**
6 Maintenance Decommissioning & Recycling	none
7 Evolution	--
8 Training & Education	--

31.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains:

AHT-EPP	**Optimization of Building Performance toolchain**
1 Requirements	**CAD software**: manual extraction of information **Microsoft Word** or **Acrobat Reader**: manual extraction of information
2 Functional Design	**Microsoft Word** and **Excel:** manual
3 Procurement & Engineering	none
4 Deployment & Commissioning	Software for programming of **data logging system:** manual or automatic read in of sensor lists (depending on manufacturer)
5 Operation & Management	**Data analysis tools, database tools: depending on manufacturer party manual, partly automated**
6 Maintenance Decommissioning & Recycling	none
7 Evolution	--
8 Training & Education	--

31.4. Gap Analysis

31.4.1. Level of automation of the toolchain & Information passed

Currently, there is no automation at all within the optimization procedure of a building and its building services.

The goal of the project is to develop a building tracker that uses measured data from the existing building services to run a building and system simulation in real-time and to feedback optimized control settings to the building management system. That would make a monitoring campaign as described in the baseline obsolete.

The information is passed manually between EPP in the baseline scenario.

Within the project, it is planned to use information from the planning phase of the building and its services to be imported in the simulation model of the building. The developed building tracker can then use this model to track the building performance.

31.4.2. Level of integration with the Arrowhead Framework

No integration with the Arrowhead framework in the baseline scenario

31.4.3. Missing Tools within the UC

The existing building automation system is planned to be coupled with a simulation software that simulates both the building envelope and the building services. To do that, a so-called building tracker will force the simulation to match the building status in real-time using measured data from the building automation system. The building tracker includes two existing tools (the IDA modeler and the IDA solver) that are part of the simulation software IDA ICE.

Another tool under development is the occupancy and behavior tracker which uses measured data from the building to generate inputs about user occupancy and behavior for the simulation program.

To couple the building tracker with measured data from the building, an OPC UA server will be used.

Both the building tracker and the occupancy and behavior tracker are mandatory for the use case.

31.4.4. Missing Tools Outside the UC

N/A

31.5. Declared Tools

Name of the tool	This is a tool in the following phase:	Compatible with AHF	Inputs	Outputs
OPC UA Server	EPP 4, 5	yes ??	Signals from Building Automation System (BAS) Optimized control signals from the Building Tracker	Measured values to Occupancy and Behavior Tracker and Building Tracker

Occupancy and Behavior Tracker	EPP 3, 4, 5	Is planned to be compatible with AHF	Signals from the **OPC UA server**	Values about user occupancy to **Building Tracker**
Building Tracker	EPP 3, 4, 5	Is planned to be compatible with AHF	Signals from the **OPC UA server** and from the **Occupancy and Behavior Tracker**	Building performance data, predictions and virtual sensors to user interface Optimized control signals to **OPC UA server**

32. UC-18 [Boliden, LTU, BnearIT]: Secure sharing of IoT generated data with partner ecosystem

Contact: Markus Frank [markus.frank@boliden.com]

The description of each Use Case can be found in D1.2

32.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 3

AHT-EPP	Data source integration into data platform	Security setup for data source	Data provisioning
1 Requirements	Prioritized list of data sources to be included is created and maintained by SH2.	For each data source in scope the SH2 is aligning security requirements with SH1. SH2 checks if prerequisites are available to set security in data platform	Requirements for data provisioning are described by SH3. SH2 assesses need and decides on the appropriate way to make the data set available.
2 Functional Design	SH2 iterates with SH1 the technical and functional details to load data into platform	SH2 applies security concepts to defined parameters. If needed the data is enriched to ensure the efficient provision of data	SH2 designs the provisioning of the data set.
3 Procurement & Engineering	Based on parameters the data source is integrated by SH2.	handled as part of data source integration into data platform	SH2 implements provisioned data
4 Deployment & Commissioning	The deployment & commissioning is made SH2. Verification of data quality by SH1	handled as part of data source integration into data platform	SH2 implements and test fulfillment in delivery according to spec, quality and security
5 Operation & Management	Handled by SH2 as part of ongoing service management, Voliden has a model similar to the PM3 system support model	handled as part of data source integration into data platform	handled by SH2
6 Maintenance Decommissioning & Recycling	Handled by SH2 as part of ongoing service management, Voliden has a model similar to the PM3 system support model	handled as part of data source integration into data platform	handled by SH2. Regular review for usage performed
7 Evolution	Handled by SH2 in alignment with feedback from SH1 and SH3	Security setup itself and changed security requirements are managed in regular service management meeting and included in release if needed	Handled by SH2 in alignment with feedback from SH3 and SH1 (for changes coming from data sources)
8 Training & Education	SH2 created needed documentation for data sources, metadata and interpretations if needed. Education for SH1 is done data source by data source	handled as part of data source integration into data platform	SH2 educated data users on data usage possibility, limitation and duties

32.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 3

AHT-EPP	Component 1 toolchain	Component 2 toolchain	Component 3 toolchain
1 Requirements	Microsoft office suite, Azure DevOps	Microsoft office suite, Azure DevOps	Microsoft office suite, Azure DevOps
2 Functional Design	Microsoft office suite, Azure DevOps	Microsoft office suite, Azure DevOps	Microsoft office suite, Azure DevOps
3 Procurement & Engineering	Azure DevOps and data components in reference architecture	Azure DevOps and data components in reference architecture	Azure DevOps and data components in reference architecture
4 Deployment & Commissioning	ServiceNow for IT Service management	ServiceNow for IT Service management	ServiceNow for IT Service management
5 Operation & Management	ServiceNow for IT Service management	ServiceNow for IT Service management	ServiceNow for IT Service management
6 Maintenance Decommissioning & Recycling	ServiceNow for IT Service management	ServiceNow for IT Service management	ServiceNow for IT Service management
7 Evolution	Microsoft office suite, Azure DevOps	Microsoft office suite, Azure DevOps	Microsoft office suite, Azure DevOps
8 Training & Education	Microsoft office suite, Azure DevOps	Microsoft office suite, Azure DevOps	Microsoft office suite, Azure DevOps

32.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 3

AHT-EPP	Data source integration into data platform	Security setup for data source	Data provisioning
1 Requirements	Manual based on standards, automation partly supported by Azure DevOps	Manual based on standards, automation partly supported by Azure DevOps	Manual based on standards, automation partly supported by Azure DevOps
2 Functional Design	Manual based on standards, automation partly supported by Azure DevOps	Manual based on standards, automation partly supported by Azure DevOps	Manual based on standards, automation partly supported by Azure DevOps
3 Procurement &	Manual based on standards, automation partly supported by	Manual based on standards, automation partly supported by	Manual based on standards, automation partly supported by

Engineering	Azure DevOps	Azure DevOps	Azure DevOps
4 Deployment & Commissioning	Manual based on standards, automation partly supported by Azure DevOps	Manual based on standards, automation partly supported by Azure DevOps	Manual based on standards, automation partly supported by Azure DevOps
5 Operation & Management	Manual based on standards, automation partly supported by Azure DevOps	Manual based on standards, automation partly supported by Azure DevOps	Manual based on standards, automation partly supported by Azure DevOps
6 Maintenance Decommissioning & Recycling	Manual based on standards, automation partly supported by Azure DevOps	Manual based on standards, automation partly supported by Azure DevOps	Manual based on standards, automation partly supported by Azure DevOps
7 Evolution	Manual based on standards, automation partly supported by Azure DevOps	Manual based on standards, automation partly supported by Azure DevOps	Manual based on standards, automation partly supported by Azure DevOps
8 Training & Education	Manual based on standards, automation partly supported by Azure DevOps	Manual based on standards, automation partly supported by Azure DevOps	Manual based on standards, automation partly supported by Azure DevOps

32.4. Gap Analysis

32.4.1. Level of automation of the toolchain & Information passed

With the legacy approach no standardization and automation of security-related processes was possible. The use case will provide prerequisites such as what data quality, attributes need to be available for time-series data and data ownership. But also implement the security concept to also efficiently share data while maintaining security in place. The time from request to data available for decision making will be drastically reduced.

Since the use case is on the interoperability level the information passed from one engineering phase to the next is the information itself and meta information to enable processing at next stage.

In the legacy approach the exchange if information was not formalized and handled differently case by case with high risk of arrows. The platform and its data provision process support structured data and metadata management as well as enable automation that LTU is looking at.

With focus on the data provisioning the two parts are the data source integration into the data platform and then the actual provisioning to data users in or via the platform.

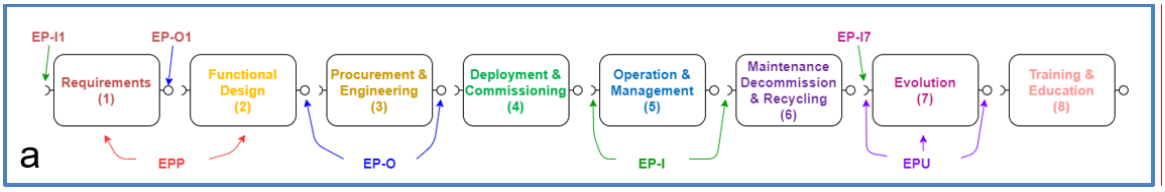
In the requirements phase the information passed to functional design is the data source to be integrated and key information to correctly set up the source technically but also to allow proper access governance. This includes the technical details on the source, how to get the data, contact persons and depending on the state of the time series based data also information for

cleansing and interpretation, e.g. what does “0” or a missing value mean. Important is also information on data ownership and access considerations. This is handled manually

From functional design to procurement the technical information alongside security requirements are considered and then developed and included in the release as part of deployment and commissioning. This means the data is included in the data platform and security is set up according to the security parameters defined. In operations management then the data set can be provisioned and information flowing from deployment and commissioning are built into the platform to allow the efficient preparation of data sets.

The information itself contains the time-series based data and a unique ID. Which is then linked to asset ID’s, data owners and security parameters.

32.4.2. Level of integration with the Arrowhead Framework



On interoperability level the interaction with arrowhead framework can be with gateway services (see Boliden UC-13). With LTU we are looking into automatic access management which will bring in the arrowhead framework aspect as well.

32.4.3. Missing Tools within the UC

Related to security management, the missing tools are scalable tools to manage access management without scaling into unrewarded and complexity that cannot be handled with reasonable effort as well as automation of access control and provisioning.

Azure data platform and data/ analytics related components

32.4.4. Missing Tools Outside the UC

Data cataloging to manage related information (currently under review)

32.5. Declared Tools

Name of the tool	This is a tool in the following phase:	Compatible with AHF	Inputs	Outputs

Boliden Data Platform, including Azure DevOps	1 Requirements 2 Functional Design 3 Procurement & Engineering 4 Deployment & Commissioning 5 Operation & Management 6 Maintenance Decommissioning & Recycling 7 Evolution	use case is on interoperability level, but compatible with gateway system	Source data from OsiPi or directly production related sensors and systems	Data sets and data warehouses
OsiPi	4 Deployment & Commissioning 5 Operation & Management 6 Maintenance Decommissioning & Recycling	use case is on interoperability level, but compatible with gateway system	Source data from production sensors or databases	Master data enriched and potentially aggregated data
PowerBI	5 Operation & Management	N/A	Data sets from Boliden data platform	Reports / visualization

33. UC-19 [3E, ALLTalk, Sirris]: Deployment and configuration

Contact: Jérôme Genot [Jerome.genot@3e.eu]

The description of each Use Case can be found in D1.2.

33.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 4

AHT-EPP	SynaptiQ Configurator	Synaptiq Mediation	SynaptiQ Data Engine	SynaptiQ Operational
1 Requirements				
2 Functional Design				
3 Procurement & Engineering				
4 Deployment & Commissioning	Customer configures its PV plant Model in this module.	Based on file format and logger type/brand, the SynaptiQ Mediation Module XML file is updated.	Automatic deployment based on configuration of SynpatiQ Configurator trigger by our engineer operational team	Fully automatic based on model in SynaptiQ Data Engine
5 Operation & Management				
6 Maintenance Decommissioning & Recycling	In case of changes on site, the customer applies the same changes in the PV plant model	Based on file format and logger type/brand, the SynaptiQ Mediation Module XML file is updated.	Automatic re-deployment based on configuration of SynpatiQ Configurator trigger by our engineer operational team	Fully automatic based on model in SynaptiQ Data Engine
7 Evolution				
8 Training & Education				

33.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 4

AHT-EPP	SynaptiQ Configurator	Synaptiq Mediation	SynaptiQ Data Engine	SynaptiQ Operational
1 Requirements				

2 Functional Design				
3 Procurement & Engineering				
4 Deployment & Commissioning	component itself	**XML**: configuration of the SynaptiQ Mediation Module on how to read the raw data files. **Mercurial**: version control repository **Rundeck**: easy access scripting tool		
5 Operation & Management				
6 Maintenance Decommissioning & Recycling	component itself	**XML**: configuration of the SynaptiQ Mediation Module on how to read the raw data files. **Mercurial**: version control repository **Rundeck**: easy access scripting tool		
7 Evolution				
8 Training & Education				

33.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 4

AHT-EPP	SynaptiQ Configurator	Synaptiq Mediation	SynaptiQ Data Engine	SynaptiQ Operational
1 Requirements				
2 Functional Design				
3 Procurement & Engineering				
4 Deployment & Commissioning	component itself: manual	**XML**: manual. **Mercurial**: manual. **Rundeck**: manual.	semi-automatic (must be manually triggered)	automatic
5 Operation & Management				

6 Maintenance Decommissioning & Recycling	component itself: manual	**XML**: manual. **Mercurial**: manual. **Rundeck**: manual.	semi-automatic (must be manually triggered)	automatic
7 Evolution				
8 Training & Education				

33.4. Gap Analysis

33.4.1. Level of automation of the toolchain & Information passed

Currently the level of automation of the toolchains is restricted to the communication between the SynaptiQ Data Engine and the SynaptiQ Operational Module.

The information are passed manually through a ticketing system.

33.4.2. Level of integration with the Arrowhead Framework

Currently, there is no integration of the toolchain with the Arrowhead Framework

33.4.3. Missing Tools within the UC

Sirris will develop an anomaly detection to detect any configuration issues within the SynaptiQ Configurator or the SynaptiQ Mediation Module.

3E will develop a new Mediation Module Configurator that will communicate with all Modules.

33.4.4. Missing Tools Outside the UC

N/A

33.5. Declared Tools

Name of the tool	This is a tool in the following phase:	Compatible with AHF	Inputs	Outputs
SynaptiQ Configurator	4,6	No	Manual inputs	XML snapshot of the PV plant model
Synaptiq Mediation	4,6	No	Manual inputs	None
SynaptiQ Data Engine	4,6	No	SynaptiQ Configurator XML snapshot	None
SynaptiQ Operational	4,6	No	SynaptiQ Data Engine DB model	visual information on screen

34. UC-20 [FARR, IKERLAN]: Elastic Data Acquisition System

Contact: Jon Rodriguez, [j.rodriguez@fagorarrasate.com]

The description of each Use Case can be found in D1.2

34.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 3

Deployment and commissioning:

- Manual parameterization of applications, definition of variables, granularity, …
- Manual parameterization of transport protocol.

Operation & Management:

- Manual diagnostics performed by experts whenever some failure is detected.

Maintenance:

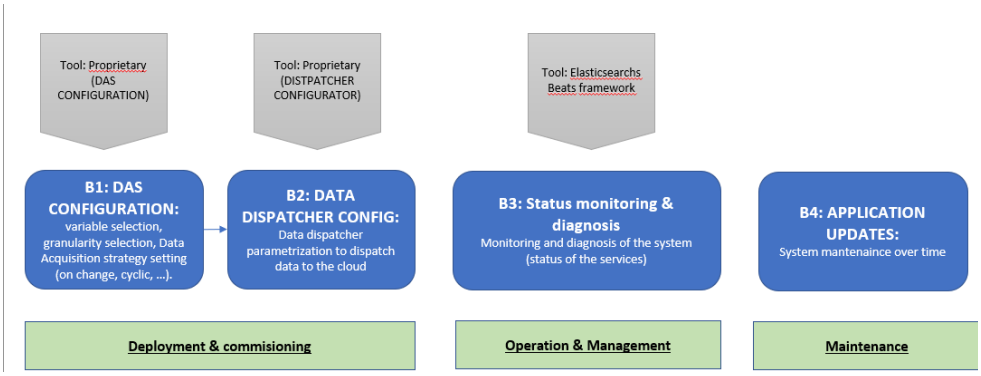
- Manual updates and deploy of the different applications.

AHT-EPP	DAS Configuration Toolchain	Monitoring and Operation Toolchain	Deployment Toolchain
1 Requirements			
2 Functional Design			
3 Procurement & Engineering			
4 Deployment & Commissioning	Fagor and Koniker develop, test, compile and install the Toolchain Ikerlan develops a Middleware to integrate the tools with Arrowhead Framework		
5 Operation & Management		Fagor and Koniker develop, test, compile and install the Toolchain Ikerlan tests the developed Middleware	
6 Maintenance Decommissioning & Recycling			Ikerlan develops the toolchain for the remote-updates.
7 Evolution			
8 Training & Education			

34.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 3

As shown in the next figure three different tools are adopted in the current UC. Two of them are inside the Deployment & Commissioning phase and one is inside Status monitoring & diagnosis phase.



- DAS configuration: User interface designed to configure the variables and their granularity.
- Dispatcher configuration: Database configuration to define the endpoints to be sent the data.
- Elasticsearch Beats framework: Within this technology, some agents are deployed within the platform with the objective of recollecting metrics and showing them through dashboards.

AHT-EPP	**DAS Configuration Toolchain**	**Monitoring and Operation Toolchain**	**Deployment Toolchain**
1 Requirements		Notepad: write the apps and thresholds	
2 Functional Design	GitLab: SW versioning system	GitLab: SW versioning system	
3 Procurement & Engineering	Visual Studio: develop .Net tools	Visual Studio: develop .Net tools	
4 Deployment & Commissioning	Fortinet/TightVNC Viewer: deploy apps in machines Middleware: Eclipse and node	Fortinet/TightVNC Viewer: deploy apps in machines Middleware: Eclipse and node	
5 Operation & Management	Visual Studio: develop .Net tools Middleware: Eclipse and node	Visual Studio: develop .Net tools Middleware: Eclipse and node	

6 Maintenance Decommissioning & Recycling			Visual Studio: develop python tools Docker: update, deploy and run containers. GitLab: SW versioning system
7 Evolution			
8 Training & Education	GitLab: SW versioning system	GitLab: SW versioning system	

34.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 3

Automation level:

- **Deployment & Commissioning:**

The variables, granularities, database configuration and so on is done by a system engineer on Fagor Arrasate's premises, prior to sending the machine to the client's plant. Once there, the set-up engineer is responsible for the final set-up and project reception, so the final configurations are done directly on the client's factory.

- **Operation & Management:**

The monitoring is currently being done automatically, shipping logs from the different elements of the value-chain to the cloud. these logs can be accessed through different dashboards in our application.

- **Maintenance:**

Nowadays, the system engineer manually deploys the applications and does the needed configuration for them to work. He is also responsible for updating the deployed versions.

Relation between engineering phases:

Connection/relation between engineering phases of
Deployment & Commissioning and *Operation & Management*: When the PLC variables and the communication protocols are defined the system will start monitoring the machine. In that moment it is necessary to make the diagnosis of the status of the system, thus, there is a linear connection between both of them (*Deployment & Commissioning* and *Operation & Management*). In the same way, if a new variable is added in the *Deployment & Commissioning* phase, this one needs to be analyzed in the *Operation & Management* phase.

Something similar occurs with the *Maintenance* phase, every issue detected on the *Operation & Management* makes to change the *Deployment & Commissioning* phase. Thus, all the engineering phases somehow are connected between them.

AHT-EPP	DAS Configuration Toolchain	Monitoring and Operation Toolchain	Deployment Toolchain
1 Requirements		Notepad: manual	
2 Functional Design	GitLab: automatic	GitLab: automatic	
3 Procurement & Engineering	Visual Studio: manual	Visual Studio: manual	
4 Deployment & Commissioning	Fortinet/TightVNC Viewer: manual Eclipse: manual	Fortinet/TightVNC Viewer: manual Eclipse: manual	
5 Operation & Management	Visual Studio: manual Eclipse: manual	Visual Studio: manual Eclipse: manual	
6 Maintenance Decommissioning & Recycling			Visual Studio: manual Docker: semi-automatic GitLab: automatic
7 Evolution			
8 Training & Education	GitLab: automatic	GitLab: automatic	

34.4. Gap Analysis

34.4.1. Level of automation of the toolchain & Information passed

The set-up of the applications is completely manual and it is not being considered the result of the monitoring to correctly do the set-up. The most interesting thing would be to automatically adjust some performance parameters based on the monitoring results.

The information is passed by files and databases. It would be nice to have a REST API in order to perform an operation between the engineering processes.

34.4.2. Level of integration with the Arrowhead Framework

Currently we are using Ikerlan's Framework. The framework is similar to Arrowhead Framework but now we are migrating to Arrowhead Framework.

Ikerlan's framework is a wrapper that uses the Arrowhead framework for the communication between the different services

34.4.3. Missing Tools within the UC

Every described tool is indispensable, thus they are different among them and they can work indistinctly. Moreover, the most important and critical tools would be the following ones:

- Monitoring and diagnosis tool.
- Deployment configuration.
- Remote updates

34.4.4. Missing Tools Outside the UC

N/A

34.5. Declared Tools

Name of the tool	This is a tool in the following phase:	Compatible with AHF	Inputs	Outputs
NT3. newTool: Get OnPremise deployed applications and versions.	Final User	Not at the moment but it will be compatible	List of applications to scan and thresholds	The list of application with the thresholds and if the apps are running
NT4. newTool: Get the status of each application (%CPU, %MEMORY, %NETWORK)	Final User	Not at the moment but it will be compatible	The list of apps and if they are running	The % of CPU and RAM in use
NT5. newTool: Calculate a proposal with the optimum parameterization	Final User	Not at the moment but it will be compatible	List of application and values: Started, CPU %, RAM and thresholds	List of applications that need to be launched or restarted
NT6. newTool: Automatic reconfiguration of the applications based on the proposal.	Final User	Not at the moment but it will be compatible	List of applications that need to be launched or restarted	No output
NT7. Getting available versions for updating	Final User	Not at the moment but it will be compatible	List of applications that are installed	List of applications available for updating
NT8. Selecting applications to be updated	Final User	Not at the moment but it will be compatible	List of applications available for updating	List of applications to be updated
NT9. Verifying compatibility	Final User	Not at the moment but it will be compatible	List of applications to be updated	Compatibility (yes/no)
NT10. Verifying system requirement and download applications	Final User	Not at the moment but it will be compatible	List of applications to be updated	Compatibility (yes/no)
NT11. Validation of new SW (simulation)	Final User	Not at the moment but it will be compatible	List of tasks for testing	Validation successful (yes/no)
NT12. Installation of applications.	Final User	Not at the moment but it will be compatible	List of applications to be installed	Installation successful (yes/no)

35. UC-21 [ABB, CSC, Wapice, VTT]: Data-based digital twin for electrical machine condition monitoring

Contact: Jan Westerlund, Peter Råback, Laurentiu Barna, Janne Keränen, Jari Halme [jan.westerlund@fi.abb.com, peter.raback@csc.fi, laurentiu.barna@wapice.com, janne.sami.keranen@vtt.fi, jari.halme@vtt.fi]

The description of each Use Case can be found in D1.2

35.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 3

AHT-EPP	Induction motor	Permanent magnet	Test bench
1 Requirements	StkH 1 defines electromechanical design requirements.	StkH 1 defines requirements for electromechanical design.	StkH 1 defines requirements for electromechanical design.
2 Functional Design	StkH 1 does the design.	StkH 1 does the design.	StkH 1 does the design.
3 Procurement & Engineering	StkH 1 does the engineering design and 2D and 3D models.	StkH 1 does the engineering design and 2D and 3D models.	StkH 1 does the bench engineering and procurement .
4 Deployment & Commissioning			StkH 1 does the commissioning and share the design for all the other StkHs
5 Operation & Management	StkH 1 does video of the machines for bearing/noise analytics. Data (*.MOV and *.MP4 files) delivered to StkH5.		
6 Maintenance Decommissioning & Recycling			
7 Evolution	Models from StkH 1 are processed by StkH 4 for modeling the losses with ML tools. Video data analyzed by StkH 5 for noise based analytics	Models from StkH 1 are processed by StkH 4 for modeling the losses with ML tools (NN & gradient boosting). StkH4 uses the models together with StkH 3 for torque estimation. In parallel StkH1 and StkH6 develop analytical models for hybrid analytics.	Based on design data StkH 2 design service architecture and dashboard for IoT, and implement it based on IoT-Ticket. StkH 2 deliver WRM24/7+ edge and connection device to StkH1. StkH 1 & 4 start planning for advanced orchestration of required services
8 Training & Education			

35.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 3

AHT-EPP	Induction motor toolchain	Permanent magnet toolchain	Test bench toolchain
1 Requirements	**Microsoft Excel + Word + Acrobat reader**: Documents with the requirements	**Microsoft Excel + Word + Acrobat reader:** Documents with the requirements	**Microsoft Excel + Word + Acrobat reader:** Documents with the requirements
2 Functional Design	**Elmer:** Electrical motor finite element modeling and simulation **Python:** Electrical motor operation and condition modeling with machine learning (ML) tools. **Python**: Digital twin construction for run time applications **Matlab:** Design of algorithms for noise analytics. **FCSMEK** tool for standard electrical machine development. **DAKOTA** for electrical machine design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis.	**Elmer:** Electrical motor finite element modeling and simulation **Python:** Electrical motor operation and condition modeling with machine learning (ML) tools. **Python**: Digital twin construction for run time applications **Matlab:** Design of algorithms for noise analytics. **FCSMEK** tool for standard electrical machine development. **DAKOTA** for electrical machine design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis.	
3 Procurement & Engineering			
4 Deployment & Commissioning			
5 Operation & Management	**Python:** Operation data analytics, executable digital twins. **Matlab:** Operation data analytics.	**Python:** Operation data analytics, executable digital twins. **Matlab:** Operation data analytics.	**Eclipse Arrowhead** tools Tools for IoT service consumers and providers. **IoT Ticket** platform for remote monitoring and management of data and responses. **IoT Ticket** dashboard. Acrobat Reader: analysis of PDF reports generated by StkH1.
6 Maintenance Decommissioning & Recycling	**Python:** Operation and maintenance data analytics, executable digital twins. **Matlab:** Operation and maintenance data analytics.	**Python:** Operation and maintenance data analytics, executable digital twins. **Matlab:** Operation and maintenance data analytics.	**IoT Ticket dashboard. Acrobat Reader**: analysis of PDF reports generated by StkH1.
7 Evolution			

8 Training & Education			

35.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 3

Mainly separated tools. The current status is that manual work is needed in between the phases, although separate sub-phase activities are mainly automated. In the future, the objective is to integrate automatically different sub-phases in order to get economic benefit

AHT-EPP	Induction motor toolchain	Permanent magnet toolchain	Test bench toolchain
1 Requirements	**Microsoft Excel + Word + Acrobat reader**: Manual connection	**Microsoft Excel + Word + Acrobat reader**: Manual connection	**Microsoft Excel + Word + Acrobat reader**: Manual connection
2 Functional Design	**Elmer:** Semiautomatic utilization Elmer as a service (REST-API). **S**imulation semiautomatic connection to python ML. **Python:** Semiautomatic connection to Elmer simulations. **Python**: semiautomatic run time applications generation **Matlab:** Manual. **FCSMEK** Manual. **DAKOTA** Manual.	**Elmer:** Semiautomatic utilization Elmer as a service (REST-API). Simulation semiautomatic connection to python ML. **Python:** Semiautomatic connection to Elmer simulations. **Python**: semiautomatic run time applications generation **Matlab:** Manual. **FCSMEK** Manual. **DAKOTA** Manual.	
3 Procurement & Engineering			
4 Deployment & Commissioning			
5 Operation & Management	**Python:** Operation data automatic analytics **Matlab:** Operation data semiautomatic analytics.	**Python:** Operation data automatic analytics **Matlab:** Operation data semiautomatic analytics.	**Eclipse Arrowhead** service (consumers and providers) dynamic orchestration. **IoT Ticket** automatic utilization of connected data nodes (WRM24/7+ and other measurements). **IoT Ticket** dashboard automatic data and KPI visualization. Automatic generation of PDF reports
6 Maintenance Decommissioning & Recycling	**Python:** Semiautomatic operation and maintenance data analytics. **Matlab:** Semiautomatic operation and maintenance data analytics.	**Python:** Semiautomatic operation and maintenance data analytics. **Matlab:** Semiautomatic operation and maintenance data analytics.	**IoT Ticket dashboard. Acrobat Reader**: Automatic generation of reports

7 Evolution			
8 Training & Education			

35.4. Gap Analysis

35.4.1. Level of automation of the toolchain & Information passed

Currently driving/automation and maintenance related data are mainly manually integrated. Driving/operating parameters and performance data/information mainly automatically integrated at the system level.

The aim is that in the project, engineering data will be better integrated organically with electrical motor operation, operation/performance management and maintenance phases. At the moment this is done mostly manually/separately based on the adopted practices and related stakeholders. The information type consists of documentation, simulation and testing data matrices, services (operation management and maintenance) and file transfer. e.g. *.csv, *.mat, *.json files, HTTP, ..

35.4.2. Level of integration with the Arrowhead Framework

At the project start, the Arrowhead Framework (AF) and its services were not in use at the use case level. During the project there will be developed functionalities/practicalities targeted into engineering, operation and maintenance phases of the electrical motors and their digital representatives, twins. From the digital part of the functionalities, the tools will be servitized when relevant. The servitized tools will be from the suitable parts based on AF compliant, registered services (both consumers and providers) at the local level over different AF compliant systems.

35.4.3. Missing Tools within the UC

Link from the design phase to performance evaluation. Improved electrical motor performance and condition evaluation models. Automatic service, simulation and model update and advanced orchestration.

Indispensable tools: Simulator, physical electrical motor and inverter/driver, distributed control system, machine learning tool and analytical tool environment, hybrid solution, digital twin, performance evaluation, dynamic modeling and model update, failure prediction

35.4.4. Missing Tools Outside the UC

N/A

35.5. Declared Tools

Name of the tool	This is a tool in the following phase:	Compatible with AHF	Inputs	Outputs

Induction motor	<StkH1 - EPP2> <StkH3 - EPP2> <StkH4 - EPP2> <StkH5 - EPP2>	No	<StkH1 - EPP2> Design data, simulation data from <StkH3 - EPP2> <StkH3 - EPP2> Output from mesh creation tools, measurements. <StkH4 - EPP2> and <StkH5 - EPP2>. Output from <StkH1 - EPP2> and <StkH3 - EPP2>	<StkH1 - EPP2> Responses <StkH3 - EPP2> Simulation responses <StkH4 - EPP2> and <StkH5 - EPP2>. Analytics and algorithms for digital twins
Permanent magnenet motor	<StkH1 - EPP2> <StkH3 - EPP2> <StkH4 - EPP2> <StkH5 - EPP2>	No	<StkH1 - EPP2> Desing data, simulation data from <StkH3 - EPP2> <StkH3 - EPP2> Output from mesh creation tools, measurements. <StkH4 - EPP2> and <StkH5 - EPP2>. Output from <StkH1 - EPP2> and <StkH3 - EPP2>	<StkH1 - EPP2> Responses <StkH3 - EPP2> Simulation responses <StkH4 - EPP2> and <StkH5 - EPP2>. Analytics and algorithms for digital twins
Test bench	<StkH2 - EPP 5, 6>	Yes	<StkH1 - EPP2> Test bench measurements (noise and vibration acceleration)	Analytics, visualization, reports of operation KPIs and maintenance needs

36. UC-22 [STM, TECHNE, Magillem, CEA, LTU, BME]: Eclipse Arrowhead training tool

Contact: Marcello Coppola, Saadia Dhouib, Emmanuel Vaumorin [marcello.coppola@st.com, saadia.dhouib@cea.fr, vaumorin@magillem.com]

The description of each Use Case can be found in D1.2

36.1. Engineering Actions for each toolchain (Baseline)

Number of Toolchains: 1

AHT-EPP	Component 1 (Papyrus)	Component 2 (Studio3Education/Blockly)	Component 3 (STM32 cards)
1 Requirements	Requirements are modeled in Papyrus using SysML requirements diagrams.		
2 Functional Design	The functional Design of the application is modeled in Papyrus using the Arrowhead SysML profile.		
3 Procurement & Engineering		The development of the application is done graphically in Studio4Education/Blockly tool.	
4 Deployment & Commissioning			The deployment of the code to the embedded STM32 cards is done
5 Operation & Management			
6 Maintenance Decommissioning & Recycling			
7 Evolution			
8 Training & Education			

36.2. Adopted Tools for each toolchain (Baseline)

Number of Toolchains: 1

AHT-EPP	Component 1 (Papyrus) toolchain	Component 2 toolchain (Studio3Education/Blockly)	Component 3 (STM32 cards) toolchain
1 Requirements	Papyrus		
2 Functional Design	Papyrus		
3 Procurement & Engineering		Studio4Education/Blockly	
4 Deployment & Commissioning			STM32 MCU and STM32Duino development environments: STM32Cube and CliArduino
5 Operation & Management			
6 Maintenance Decommissioning & Recycling			
7 Evolution			
8 Training & Education			

36.3. Level of Integration of each toolchain (Baseline)

Number of Toolchains: 1

AHT-EPP	Component 1 (Papyrus) toolchain	Component 2 (Studio3Education/Blockly) toolchain	Component 3 (STM32 cards) toolchain
1 Requirements	requirements models are connected to functional design models using traceability links.		
2 Functional Design	Functional design models are connected to the engineering tool through automatic code generation		
3 Procurement & Engineering		Studio4Education is configured by the json files exported from Papyrus. Studio4Education automatically generates C code including STM32 libraries	
4 Deployment &			C Code provided by

Commissioning			Studio4Education is uploaded to the STM32 cards. During the application execution, actuators and sensors connected to STM32 boards send real time data to Studio4Education
5 Operation & Management			
6 Maintenance Decommissioning & Recycling			
7 Evolution			
8 Training & Education			

36.4. Gap Analysis

36.4.1. Level of automation of the toolchain & Information passed

Papyrus and STudio4Education are communicating automatically via code generation. JSON code is generated from Papyrus and consumed by Studio4Education. Specification are for this JSON file is read from Studio4Education settings.

JSON files generated from Papyrus contain the list of actuators and sensors that will be integrated in the toolbox of STudio4Education. It also contains information of complexity for education use case.

36.4.2. Level of integration with the Arrowhead Framework

At the project start, the Arrowhead Framework (AF) and its services were not in use at the use case level. During the project, the three tools used in the use case will communicate via the Arrowhead Framework.

36.4.3. Missing Tools within the UC

We are going to make the link between the graphical language and the underline compiler of STM32

CLIDuino is indispensable for this UC.

36.4.4. Missing Tools Outside the UC

Blockly

36.5. Declared Tools

Name of the tool	This is a tool in the	Compatible with AHF	Inputs	Outputs

	following phase:			
Component 1 (Papyrus)	Phases 1 & 2	yes through the integration of the SysML Arrowhead profile and through code generation to Arrowhead	1. A textual specification of the IoT application to be developed. 2. A library of sensors/actuators coming from Studio4Education	Papyrus provides a list of actuators and sensors (json format) to Studio4Education (Blockly)
Component 2 (Studio4Education/Blockly)	Phase 3	is planned to be compatible	1. A json file from Papyrus, 2. Real-time data coming from STM32 boards	1. Studio4Education exports a C code to the STM32 MCU and STM32Duino cards, 2. Real-time data provided as service to the AHF
Component 3 (STM32 boards)	Phase 4	is planned to be compatible	C code from Sudio4Education	Actuators and sensors connected to the STM32 boards send real time data to Studio4Education

37. Revision history

37.1. Contributing and reviewing partners

Contributions	Reviews	Participants	Representing partner
X	X	Marek Tatara	DAC
X	X	Federico Montori	IUNET
	X	Géza Kulcsár	IQL
X		Lukáš Maršík	CAMEA
X		Önder Babur, Mahdi Saeedi Nikoo, Sander Thuijsman, Ferry Timmers, Alireza Mohammadkhani, Jeroen Voeten, Marc Geilen, Jan Friso Groote, Michel Reniers, Loek Cleophas	TU/e
X		Jose María Alvarez Rodríguez	UC3M
X		Ramon Schiffelers, Sven Weber	ASML
X		Koen van Wijk, Oscar Reynhout	ICTG
X		Anja Zernig	KAI
X	X	Patrick Moder, Benedikt Schnell, Fetting Timon	IFAG
X		Lars Oskarsson	LBB
X		Carlos Yurre	FAUT
X		Maurizio Griva	REPLY
X		Davide Brunelli, Matteo Zauli	IUNET
X		Edoardo Patti, Gianvito Urgese	POLITO
X		Sara Bocchio	ST-I
X		José Luis Burón	Acciona
X		Mustafa Küçükkuru, Alper Özel, Çağlar Henden	ARCELIK
X		Ralph Weissnegger, Christian Ettinger	CISC

X		Jakub Rewieński	GUT
X		Guoyuan Li	NTNU
X		Markus Frank	Boliden
X		Johannes Kristan, Michael Leippert, Valentin Fetscher	Bosch
X		Germar Schneider, Matthias Fehr, Dirk Mothes	IFD
X		Paul Patolla, Dirk Reichelt	?
X		Vahid Salehi	?
X		Mario Thron, Christian Hübner, Thomas Bangemann	IFAT
X		Dagmar Jähnig	AEE
X		Jérôme Genot	3E
X		Jon Rodriguez	FARR
X		Jan Westerlund	ABB
X		Peter Råback	CSC
X		Laurentiu Barna	Wapice
X		Janne Keränen, Jari Halme	VTT
X		Saadia Dhouib	CEA
X		Marcello Coppola	STM
X		Emmanuel Vaumorin	Magillem

37.2. Amendments

No.	Date	Version	Subject of Amendments	Author
1	2020-09-27	1.1	First draft	Federico Montori, Marek Tatara
2	2020-10-07	1.2	Insertion of new use cases	Federico Montori
3	2020-10-16	1.3	Insertion of new use cases	Marek Tatara
4	2020-11-02	1.9	Filling missing use cases	Marek Tatara
5	2020-11-02	2.0	Final Sanity Check	Federico Montori, Marek Tatara
6	2020-11-25	2.0	Internal Review	Patrick Moder, Benedikt Schnell, Fetting Timon

7	2020-12-02	2.0	Internal Review	Laurentiu Barna
8	2020-12-03	2.1	Addressing comments of the internal reviewers	Federico Montori

37.3. Quality assurance

No	Date	Version	Approved by
1	2020-12-03	2.1	Jerker Delsing