Document title: Arrowhead Tools best practices

Version	Status	Date
1.0	final	2019-12-10

Author
Federico Montori

Contact
federico.montori2@unibo.it

Best practices

IUNET: Federico Montori
federico.montori2@unibo.it

IQL: Géza Kulcsár
geza.kulcsar@incquerylabs.com

Abstract
This document is a collection of more user-friendly definitions as well as best practices when defining and describing tools, being them used within a Use Case or general purpose tools.

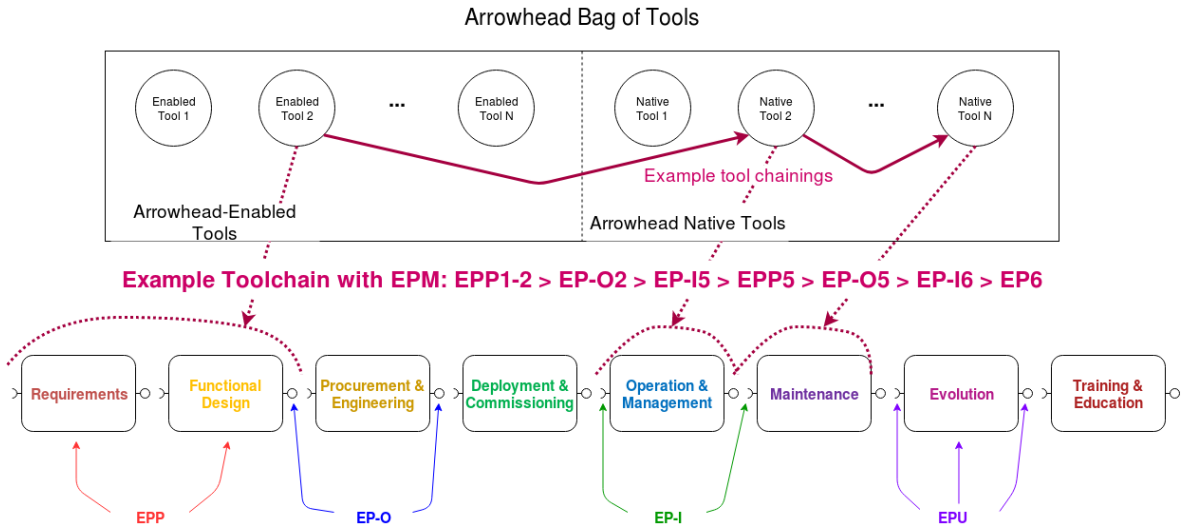
Table of contents

1. Best practices

This document is a collection of more user-friendly definitions as well as best practices when defining and describing tools, being them used within a Use Case or general purpose tools.

1.1 Common practices

Recall that the phases and the elements of the EAEM were already defined in another document, but here a short recollection can be found. Recall that EP-O and EP-I are respectively inputs and outputs that connect different phases, and not a generic input and output of the tool. It is important to state that **EPPs do not necessarily have to interact in the order presented in the figure**; that is only a suggested sequence of phases in a common understanding of an engineering process and gives us a standard numbering. That being said, recent advances imply that such a process can be iterative, skip phases or even consists of loops.



A recall to the alternative naming:

EPPs are numbered as follows:

- EPP1: Requirements
- EPP2: Functional Design
- EPP3: Procurement & Engineering
- EPP4: Deployment & Commissioning
- EPP5: Operation & Management
- EPP6: Maintenance
- EPP7: Evolution
- EPP8: Training & Education

Similarly, EP-I and EP-CI are numbered as follows:

- EP-I1: Input for Requirements
- EP-I2: Input for Functional Design
- EP-I3: Input for Procurement & Engineering

- EP-I4: Input for Deployment & Commissioning
- EP-I5: Input for Operation & Management
- EP-I6: Input for Maintenance
- EP-I7: Input for Evolution
- EP-I8: Input for Training & Education

- EP-O1: Output of Requirements
- EP-O2: Output of Functional Design
- EP-O3: Output of Procurement & Engineering
- EP-O4: Output of Deployment & Commissioning
- EP-O5: Output of Operation & Management
- EP-O6: Output of Maintenance
- EP-O7: Output of Evolution
- EP-O8: Output of Training & Education

Regardless of the internal structure defined for your use case, for the project purposes it is convenient that each partner applies the above Extended Automation Engineering Process (EAEM) for their System of Systems (SoS). This will ensure a homogenization and possible interaction between partners and SoSs. It is important to stick to such a structure even in case your SoS does not go through all the phases. In such a situation you would declare only the phases that concern your SoS.

1.2 Identify the need for a Tool

Given the formal definition of an Arrowhead Tool given in the WP4 O1 document (https://atmospheres.research.ltu.se/owncloud/index.php/f/991334), one should determine whether the developed SoS requires using one or more AH tools to achieve the intended functionality. Indeed, most time it does. In fact, besides the application systems that are strictly functional part of the use case, any software that helps you in the design your SoS, as well as plays supporting role in the run-time phase, can be an AH tool. In particular, whenever there is a **performance or data processing issue**, an AH tool can be introduced in the toolchain and its presence will reduce engineering costs of the SoS. An AH tool is also needed when there is an **interoperability or communication gap**, in such a case, the outcome would be interoperability for IoT and SoS. An AH tool can also address several other needs such as the introduction of new features to SoS.

1.3 Declare your Tool within the Toolchain Architecture

When defining an AH tool, it is important to summarize a series of defined key points, which shall help to identify the role and function of that tool within the Arrowhead Tools Project. The key points are explained in the following (examples regarding each phase, for a major clarity, can be found in the next section):

a. **Whether the tool is run-time or design-time**: an AH Tool can be design-time or run-time. More in detail, if the tool is used typically in phases that characterize the preparatory stage in the engineering process of the SoS, it is a design-time tool, whereas if the tool is used while the SoS is up and running in its steady state, it is a run-time tool.
b. **Whether the tool is connected to the AHF**: the tool is AH-compliant or connected to the AHF if it consumes the services offered by at least the three mandatory core

services: Service Discovery, Authorization and Authentication, and Orchestration. To this end, a run-time tool is highly recommended to be AH-compliant as it likely implements service interfaces that have to be used in run-time automation processes by other components of the SoS. On the other hand, depending on their purpose and usage, it might not be possible or convenient to make design-time tools AH-compliant (e.g. because they require less automation, more interaction with humans or are on a different abstraction level).

c. **Which EPU is the tool involved in**: when defining a tool it is strictly necessary to state clearly and unequivocally which EPU it refers to. The quickest and most efficient way to map a tool to EPUs is to list the relevant EPUs using the naming that we gave at the beginning of the document.
   For example: EPB3, EP-O3, EP-I4 identifies a (likely design-time) tool that is specifically devoted to the Procurement & Engineering phase and it implements also an interface between Procurement & Engineering and Deployment & Commissioning. As a guideline, we can associate conceptually the first 4 EPP with the design-time and the others with the run-time as it is a situation that is commonly applicable; this is **not** a sharp division as all the combinations are technically possible.
d. **The inputs of the tool**: it is necessary to define the input of the tool (if any). A tool can have several units of input; The definition must be structured as a list of units structured in the following way: [input data], (EP-I).
   In fact, each input, if it is a toolchain input, may be gathered by an EP-I, in such a case this must be indicated (it is mandatory only if the input is part of the toolchain, i.e. if it is obtained from another tool).
e. **The outputs of the tool**: it is necessary to define the output of the tool (if there is any). A tool can have several units of output; The definition must be structured as a list of units structured in the following way: [output data], (EP-O).
   In fact, each output, if it is a toolchain output, may be gathered by an EP-O, in such case this must be indicated (it is mandatory only if the output is part of the toolchain, i.e. if it is delivered to another tool).

1.4 Tool declaration examples

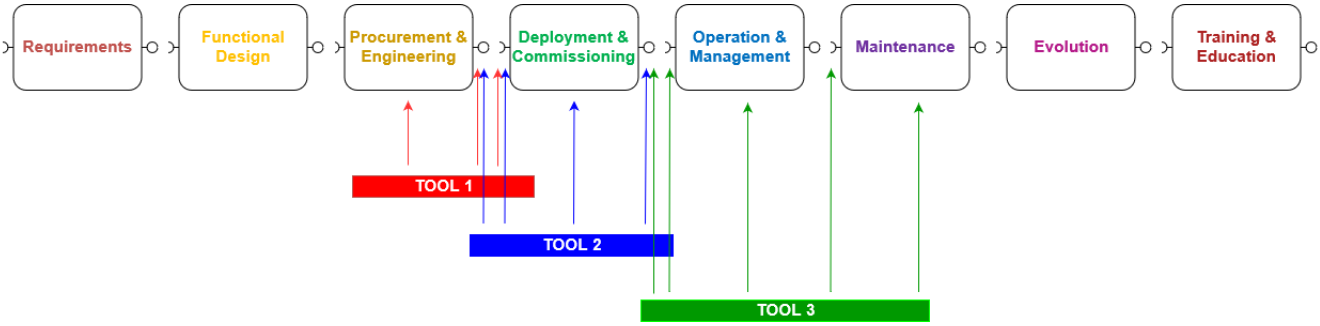
In this section we report a series of tools that can serve as examples for declaring the tools in the Arrowhead Tools project.

The Validation Toolchain

This abstract toolchain is a sandbox composed of three tools. The *validator tool* performs several performance benchmark tests on a device given in input, provided that it has certain functionalities (sensing and networking). Let's assume that such devices are used in a multitude for tasks like Structural health monitoring (where one single sensing device is not enough). This validator tool also gives the output of the test to another tool (the deployment tool) which, taken as input the outputs of the tests and the map of the environment, provides the best placement of the devices in the environment in order to optimize the connection issues. Next, we have the optimizer tool, which is a run-time tool that periodically checks the status of the sensor network and turns them off and on in order to optimize the amount of energy consumed. It needs the sensor status as well as the deployment information of them. It also checks for anomalies in the sensors and sends an alarm when detected (e.g. the battery of one of the sensors is exhausted).

1. Validator Tool
   a. Design-Time Tool

   b. Not connected to the AHF
   c. EPP3, EP-O3, EP-I4
   d. [Device Data Sheet]
   e. [Performance Test Output] (EP-O3)
2. Deployment Tool
   a. Design-Time Tool
   b. Connected to the AHF
   c. EP-O3, EP-I4, EPP4, EP-O4
      The EP-O4 is not coupled with EP-I5 either because there is no other tool in this toolchain or because the output is meant to serve whichever EPP needs it instead of a specific one (in our case both maintenance and operation).
   d. [Performance Test Output] (EP-I4)
      [Environment Description]
   e. [Device Deployment] (EP-O4)
3. Optimizer Tool
   a. Run-Time Tool
   b. Connected to the AHF
   c. EPP5, EPP6, EP-I5, EP-I6
      It concerns both automation and maintenance. It has no output for the toolchain but it consumes input from the deployment.
   d. [Sensor Deployment] (EP-I5, EP-I6)
      [Sensor Data]
   e. [Turnoff-Turnon Commands to Sensors]
      [Alarm to the Human Manager]



2. Connection to the Arrowhead Framework

Arrowhead Tools, whenever possible, should be connected to the Arrowhead Framework, which will facilitate and unify the information transition between tools belonging to different EPUs. In particular, whenever a tool implements an input or output interface (EP-I or EP-O), then such interface should make the tool an Arrowhead provider or an Arrowhead consumer. Recall that the compliance with the AHF is achieved by implementing a consumer interface against the three mandatory Core Services: Service Discovery, Authorization and Orchestration.

In fact, we can state that a fully Arrowhead-compatible toolchain makes the Arrowhead Framework itself the toolchain architecture, in which each toolchain becomes a set of Orchestration rules.

In short, the Arrowhead Framework that can be adopted as:

- an integration platform to improve the tool chains automation, and consequently the engineering process;
- an integration platform for the creation of new applications (“old” way of using the Arrowhead Framework investigated in the previous project).

3. List of abbreviations

Abbreviation	Meaning
AHF	Arrowhead Framework
SoS	System of Systems
CP SoS	Cyber-Physical System of Systems
EPU	Engineering Process Unit
EPP	Engineering Process Phase
EAEM	Extended Automation Engineering Model
EP-I	Engineering Process Input Interface
EP-O	Engineering Process Output Interface
EPM	Engineering Process Mapping

4. Revision history

4.1 Contributing and reviewing partners

Contributions	Reviews	Participants	Representing partner
X	X	Federico Montori	IUNET
X		Géza Kulcsár	IQL
	X	Marek Tatara	DAC
X		Ákos Horváth	IQL

4.2 Amendments

No.	Date	Version	Subject of Amendments	Author
1	2019-10-01	0.1	First draft	Federico Montori
2	2019-10-25	0.2	Draft for evaluation	Marek Tatara
3	2019-10-28	0.3	Compliance with AHF	Federico Montori
4	2019-12-10	1.0	Final Version	Federico Montori, Marek Tatara

4.3 Quality assurance

No	Date	Version	Approved by
1	2019-12-10	1.0	Jerker Delsing