Document title: General-Purpose Toolchains

Version	Status	Date
1.1	final	2020-12-03

Author
Géza Kulcsár

Contact
geza.kulcsar@incquerylabs.com

General-Purpose Toolchains

IQL: Géza Kulcsár
geza.kulcsar@incquerylabs.com

IUNET: Federico Montori
federico.montori2@unibo.it

Abstract

This document contains the list of toolchains that are not associated with any particular use case. Each of the tolchains is described in the context of Arrowhead compliance, place in the Engineering process model, and its components.

Table of contents

1. Introduction

The present document serves to lay the foundations of a recent concept within Arrowhead Tools, which necessity has become evident in the past months and which will receive more exposure in the second part of the project.

The idea of a toolchain, as detailed elsewhere, subsumes a specific combination of different software tools, which does not necessarily only serve a specific industrial purpose, but which is utilizable for any present and future Arrowhead installments.

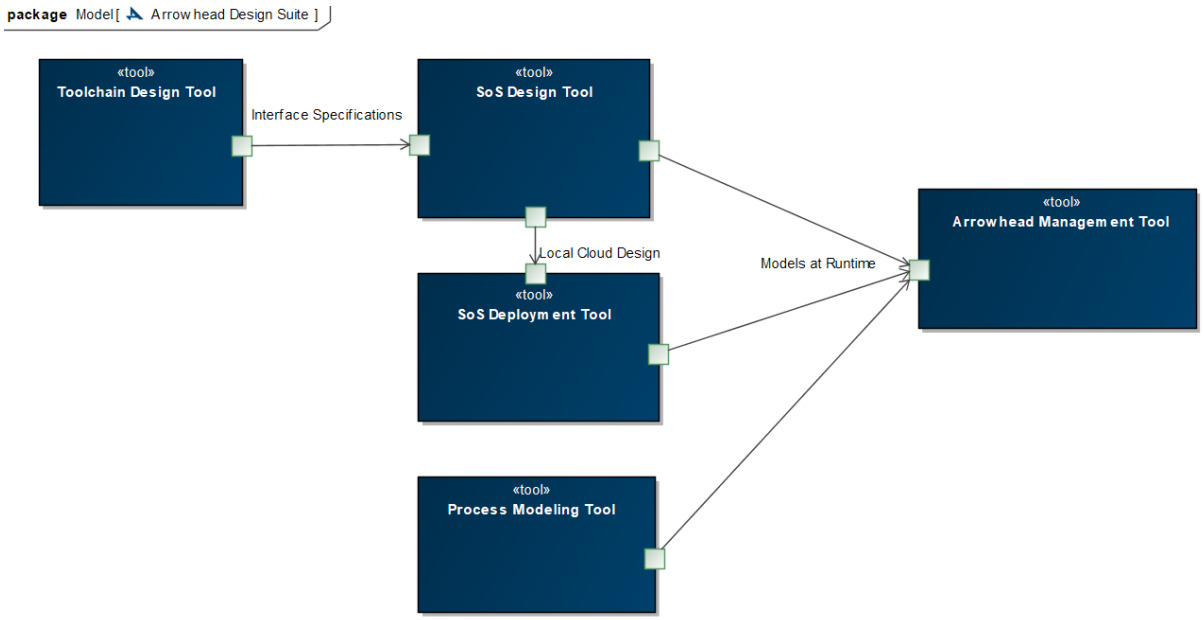
This document describes two of them, anticipating that further entries will emerge during the last milestone of the project.

2. The Arrowhead Design Suite: A General-Purpose Toolchain for SoS and Toolchain Design and Deployment

The *Arrowhead Design Suite* (ADS) is a toolchain based on systems modeling, relying on the standard, most established systems engineering language, SysML (https://sysml.org/).

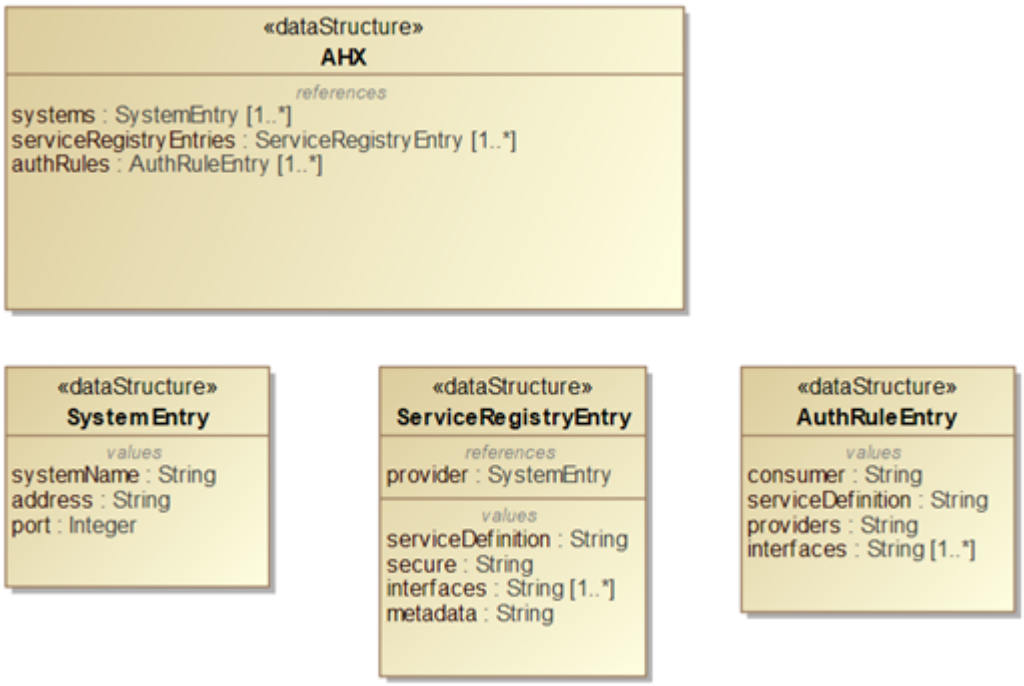
The core of ADS is a plugin collection created for *MagicDraw (*also known as *Cameo Systems Modeler)* from CATIA No Magic, arguably the most established industry-scale systems engineering tool. However, ADS provides much more than that: it makes the relevant parts of the created design models *alive* by communicating them in the right format to the *Arrowhead Management Tool*, AHT's standard operation interface.

ADS unifies different abstraction levels. The highermost abstraction, manifested as the *Toolchain Design Tool*, is there for specifying the structure of the toolchains themselves. As a particularly relevant example, the figure below depicts the architecture of ADS represented in ADS:



The most abstract element of the ADS toolchain, the *Toolchain Design Tool* (TDT), where this figure originates, is itself represented in the upper left corner of the figure.

The *Interface Specifications* arrow denotes a tool interface where data models get exchanged: while specifying a toolchain in TDT, one might reason about the data structures being sent between tools (such as the interface data structure which this very paragraph is about). If such data objects are also used within actual Arrowhead installments, this connection might be used to communicate their structure from the toolchain to the SoS level via this link. Also, in general, arrows here represent the actual chaining between the tools, i.e., the interfaces where data might flow (but does not necessarily has to in every potential scenario). As an example, see the following figure with a TDT-level data structure modeled in SysML, being used later in SoS modeling:



The further elements in the toolchain are:

- The SoS Design Tool involves the Arrowhead SysML base profile for designing local clouds (i.e., SoS models). The depicted interface (Models at Runtime) to the right is technically realized as an exporter, which transfers SoS models to the Management Tool, where those artifacts become alive as part of a living Arrowhead installment. Another interface pointing to the block below is used for extracting the deployment-relevant aspects of the SoS model - see the next bullet.
- The SoS Deployment Tool is an accompaniment to the SoS design tool, where SoS models are augmented by a SysML-based variant of Vorto information models, i.e., device digital twins based on the Eclipse Vorto project (https://www.eclipse.org/vorto/).
- The Process Modeling Tool is a recent prototype for creating BPMN sequences as Arrowhead-compliant workflow choreographies. The recipient of the data is, again, the Arrowhead Management Tool, thus serving as a central entry point towards real-life Arrowhead applications.

Those parts of the Arrowhead Design Suite that have left the prototype stage can be publicly accessed at:

https://github.com/IncQueryLabs/arrowhead-tools-magicdraw-plugins

3. Discovering Web Things as Services within the Arrowhead Framework

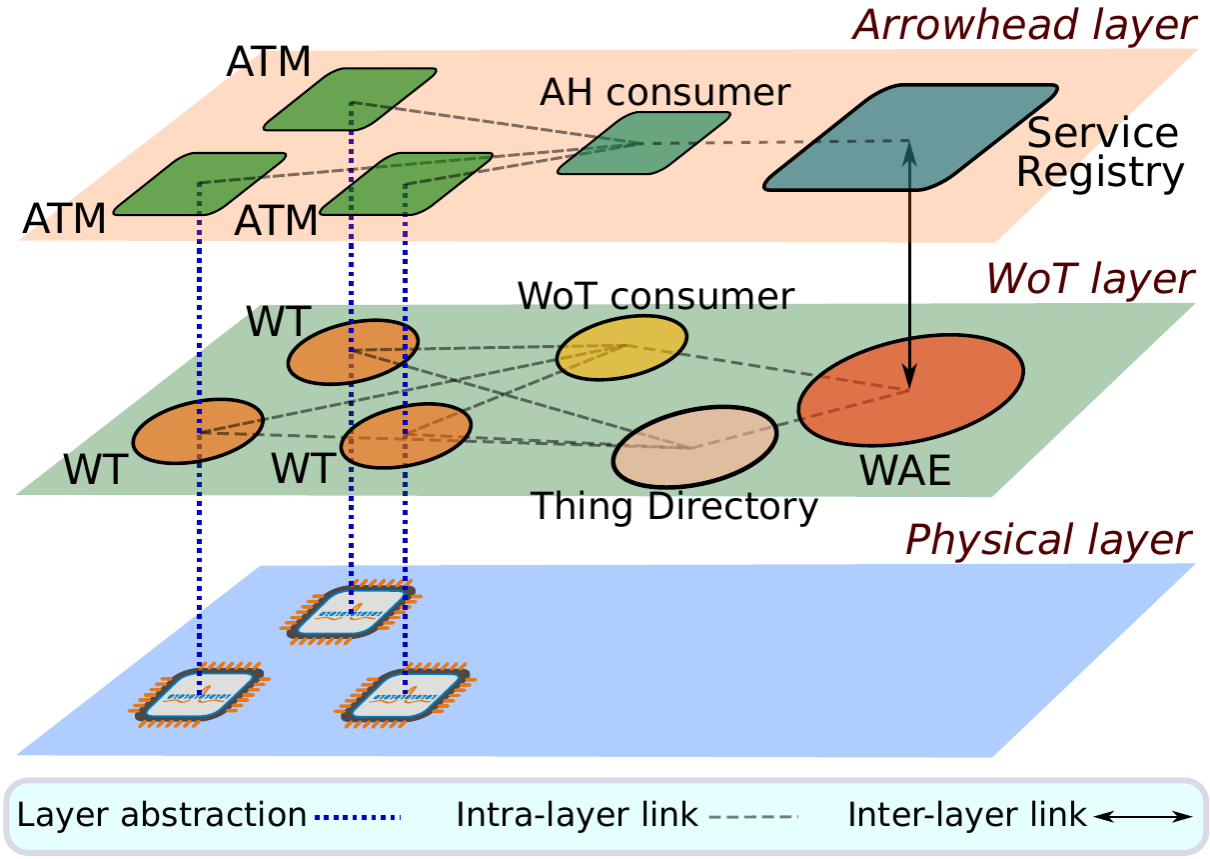
In this section we define in detail the architectural structure of our proposal for integrating Web Things within the Arrowhead Framework. The system at the basis of this architectural toolchain is the WAE (its technical details and the calls supported can be found in D3.3)

Despite the great potential of WoT paradigm, since the process of making an already existing service W3C WoT-compliant requires some effort, there can still be cases - especially those involving old legacy systems - in which a service could be interested to benefit from capabilities offered by Web Things, without joining the WoT ecosystem. In particular, our proposal is specifically designed for such components that may either be unable to understand the same language and use the same protocols as the WoT ecosystem does, or be unaware of the location of the Web Things that need to be queried.
More in detail, we envision an ecosystem in which a set of Web Things, which are devoted to collect data through sensors, expose their data to potential consumers that are external to their ecosystem, in particular:

- A consumer that is able to communicate using the WoT standard and the same communication protocol the Web Things are using, however it does not know how to reach the Web Things endpoint.
- A consumer that has no information about the Web Things endpoint and communicates only using another legacy protocol (say, HTTP).

The architectural design can be found in the figure below.

In order for the whole ecosystem or, as it is defined in the Arrowhead official documentation model (https://www.arrowhead.eu/arrowheadframework/this-is-it/documentation-model/), the System of Systems (SoS) to be able to cope with such cases, we propose an architecture in which each Web Thing is also a Service Provider of an Arrowhead local cloud, therefore exposing sensor data as a service.
The Arrowhead Framework allows indeed each service to be discoverable through its main component: the Service Registry. The Service Registry acts as the main service broker in a SOA: for each service in the local cloud that advertises its endpoint through a publish call, it keeps in memory a service record, encoded in JSON, that includes a set of service metadata. The record includes the type of service, its endpoint and the protocol used, although other metadata can be added upon need. In a typical and simple scenario, a Service Provider first publishes its service record, then a Service Consumer willing to consume such service performs a service lookup call against the Service Registry and obtains information about the endpoint and the protocol of the desired service. Once this information is held by the Service Consumer, the communication with the Service Registry is no longer needed and the Service Provider and the Service Consumer can communicate directly.



Note that in an Arrowhead service consumption the Orchestration module also has its part: it gets queried by the Service Consumer for a service of a defined type and it searches the Service Registry for the most suitable service record, based on a set of rules. The use of the Orchestration service is out of the scope of this paper, therefore we intentionally simplify the interaction bypassing the Orchestration and only demonstrating the architectural integration.

4. Revision history

4.1 Contributing and reviewing partners

Contributions	Reviews	Participants	Representing partner
X		Géza Kulcsár	IQL
X	X	Federico Montori	IUNET
	X	Ármin Závada	IQL
	X	Marek Tatara	DAC
	X	Jurasky Wiking	IFAG
	X	Laurentiu Barna	Wapice

4.2 Amendments

No.	Date	Version	Subject of Amendments	Author
1	2020-10-09	0.1	First draft	Géza Kulcsár
2	2020-10-26	0.2	Added WoT Integration Architecture	Federico Montori
3	2020-11-02	1.0	Final Sanity Check	Federico Montori, Marek Tatara
4	2020-11-19	1.0	Internal review	Jurasky Wiking
10	2020-12-02	1.0	Internal Review	Laurentiu Barna
5	2020-12-03	1.1	Addressing comments from the internal review	Marek Tatara

4.3 Quality assurance

No	Date	Version	Approved by
1	2020-12-03	1.1	Jerker Delsing