Approach for Evolving Sensing and Actuation Devices in Cyberphysical Systems Architectures

Diego C. Sales[1,2] and Leandro B. Becker[1]
[1]*Graduate Program on Automation and Systems Engineering (PPGEAS), Federal University of Santa Catarina - UFSC, Florianópolis, SC, Brazil*
[2]*Federal Institute of Amazon - IFAM, Campus Manaus Distrito Industrial, AM, Brazil*

Keywords: Evolution Approach, Systems Architecture, Sensors, Actuators, Quality Attributes.

Abstract: The constant technological evolution, in this case targeting sensing and actuation devices (S&A), causes designers to evaluate potential modifications (upgrades) in the architecture of cyberphysical systems (CPS). Including or exchanging S&A devices in the architecture is a complex activity that requires the use of a systematic approach. This paper presents the CPS Architectural Evolution Approach (CAvA), which provide means to evaluate the impacts on architecture components, requirements, and software quality characteristics for mitigating risks and possible failures on meeting design goals. CAvA proposes a set of phases and activities to guide designers in the selection, evaluation, integration, and compatibility through the analysis of software quality attributes, based on the ISO/IEC 25010. A software tool was designed to support CAvA application. A case study is presented to highlight CAvA benefits for evolving the architecture of a small-scale UAV, showing a set of possible scenarios according to predefined goals.

1 INTRODUCTION

In recent years, different approaches have been proposed to guide designers in the challenge of architectural evolution. Exchanging architecture components to meet new design requirements demands the use of a systematic approach capable of encompassing a set of phases to ensure that changes do not generate risks to other subsystems and components (Bass et al., 2013).

S&A devices are important components in the architecture of CPS applications due their ability to provide information/data from the environment/plant (physical) to the (cyber) software that performs the control by sending electrical signals to actuators (Lee and Seshia, 2016). Due technological evolution, commercial-off-the-shelf (COTS) devices have expanded their application range by using new conversion phenomena (electromechanical, electrochemical), improved technical specifications (accuracy, response time), hardware interfaces, and more recently by combining multiple functionalities in a single device.

The architecture evolution by exchanging S&A devices, enables the improvement of software quality characteristics such as performance, reliability, safety, new adjustments, functionalities and requirements to extend the system life cycle and the compliance with new requirements. However, selecting potential candidates, defining, evaluating, defined criteria, and modification analysis for architecture compatibility is not a trivial task, due to interface particularities, device types, characteristics and distinct technical specifications that are defined by each manufacturer (Mohamed et al., 2007). The exchange of one or more devices may include modifications to the system architecture and design, such as: data type, access address, addition of new devices to provide the appropriate interface, provision of compatible protocols, redistribution of available resources, and new deployment parameters.

The use of an approach that supports the evaluation and analysis in the exchange of these devices is of great value to designers because it allows identifying risks and impacts on architecture. The literature presents different methods and approaches of system design that include architecture tradeoff analysis, evaluation and COTS selection (Barbacci et al., 1998; Bass et al., 2013). These approaches are composed of a set of phases and activities that guide the team of designers during the development of the systems. However, gaps in these approaches are observed, such as: selection and evaluation of multiple COTS; architec-

Sales, D. and Becker, L.
Approach for Evolving Sensing and Actuation Devices in Cyberphysical Systems Architectures.
DOI: 10.5220/0010310303060313
In *Proceedings of the 9th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2021)*, pages 306-313
ISBN: 978-989-758-487-9

ture space exploration based on COTS; impact analysis of the architecture caused by COTS tradeoff; architecture evaluation tool with support for modeling. Thus, this paper aims to present an alternative solution, named CAvA, to addresses the observed gaps.

The CAvA approach consists in a set of phases and related activities that guide designers in the exploration of scenarios, evaluation and analysis of architecture requirements and quality attributes impacted by the change of one or more S&A devices, based on the ISO/IEC 25010(ISO/IEC, 2011). The proposal uses Model-Driven Engineering (MDE) concepts for enabling the exploration of multiple solutions, providing tool support, and guiding designers in step-by-step activities. As part of the process, the Architecture analysis & Design Language (AADL) and the Requirement Specification Language (ReqSpec) (Feiler et al., 2016) are used to drive the development of the approach with models, integrated into the Open Source AADL Tool Enviroment (OSATE) and additional plugins for Model-Based Analysis.

The rest of this paper is organized as follows: section 2 presents most relevant related works, highlighting motivations for the proposed approach. Section 3 details the phases and activities of the proposed approach. Section 4 illustrates the use of the proposed approach and related tools. The section 5 presents the conclusions and future work directions.

2 RELATED WORKS

Different authors proposed methods in the form of a set of phases and activities aimed at assisting designers in the development of systems, with attention to the architecture evolution during the system life cycle. Typically, descriptive guides on "what" should be done to evaluate the tradeoff at different levels of coverage are presented.

2.1 Architecture Tradeoff Analysis Methods

The ATAM method *Architecture Tradeoff Analysis Method (ATAM)* has been used for more than a decade to evaluate software architectures in different domains, such as automotive, financial and defense (Kazman et al., 1998). ATAM approach is compose by 8 phases: Present the ATAM to stakeholders; Present business drivers; Present architecture; Identify architectural approaches; Generate quality attribute utility tree; Analyze architectural approaches meeting goals; Brainstorm and prioritize scenarios; Analyze architectural approaches with ranking scenarios and Reporting phase with Present results.

In (Barbacci et al., 1998) an extension of the ATAM method is presented, which purpose is to provide principles for performing the analysis of tradeoff in architecture components based on the modeling of quality attributes. The quality attributes listed were: Security, performance, and availability. The authors highlight that these attributes can interact with each other, generating conflicts if changes occur.

The ALMA (Architecture Level Modifiability Analysis) method focuses on modification (Bengtsson et al., 2004). ALMA consist of five phases: (1) goal selection; (2) Software architecture description; (3) survey of the scenarios of changes; (4) evaluation of the scenarios of changes; (5) results interpretation. The author proposes three approaches to compare candidates: (1) point out the best candidates for each scenario; (2) rank each candidate scenario; (3) estimate the weight of each candidate.

The authors detail the phases of "what" should be done to perform the exchange, selection and evaluation of architecture components. However, "how" to perform the phases and activities is not presented, and it is at the discretion of the designer to define the approach and tools to support the exploration of scenarios, selection and analysis of the tradeoff in architecture components. It is suggested the use of COTS components, highlighting the possibility of use in the system architecture to enable modularity and ability to future modifications. For this, a study was conducted to identify the criteria used in the approaches of selection, evaluation and analysis of COTS.

2.2 COTS Selection Approaches

The researched approaches to COTS selection were evaluated by identifying the phases, activities and techniques involved in the selection, evaluation and analysis of COTS. In (Mohamed et al., 2007) and (Garg, 2017) systematic studies of approaches to COTS selection and evaluation are presented. As a whole, 8 approaches were listed and their main characteristics, advantages and disadvantages were presented.

PORE (Procurement-Oriented Requirement Engineering) (Ncube and Maiden, 1999) encourages an iterative process between requirements elicitation and multiple COTS selection, an evaluation tool are presented to support comparative COTS candidates;

MiHOS (Mismatch-Handling aware COTS Selection) (Mohamed et al., 2007) targets the mismatches that are encountered during the COTS selection process. UnHOS (Uncertainty Handling in Commercial

Off-The-Shelf) (Ibrahim et al., 2011) They highlight the evaluation of requirements and compatibility for integration with the system.

The DesCOTS (Description, evaluation and selection of COTS) (Grau et al., 2004) approach makes selection and evaluation based on ISO/IEC9126 quality standard in which criteria are defined based on the characteristics and sub-characteristics of the candidates.

The Guided architecture trade space explorer (GATSE) tool (Procter and Wrage, 2019) design by shopping COTS devices evaluating quality attributes declared in the AADL model. with the same hardware interface as the selected device. The ArcheOpterix is a tool for optimization of architectures evaluate properties using Evolutionary algorithms and that require users to know and to be able to quantify their preferences before searching any candidate (Aleti et al., 2009). As GATSE, ArcheOpterix considers a single COTS Tradeoff with the same hardware interface and functionality, limiting functionality suitability exploration with COTS.

In (Ross et al., 2019) proposed a tool, with Clafer modeling language, for synthesis and exploration of automotive architectures in early design stage. The tool support evaluate features, functionalities, devices and interface hardware through a set of design decision templates based on the reference architecture model. The authors do not address the tradeoff of multiple devices and incompatibilities.

The related works presented here were evaluated based on advantages and disadvantages presented in (Garg, 2017), using four methodology criteria for each approach (Mohamed et al., 2007). Additionally, four new criteria are included with focus on analysis, as presented in Table 1. This paper presents the CPS Architectural Evolution Approach (CAvA), which aims to detail a set of phases and activities that meet the presented criteria as well as the identified gaps presented in Table 1.

Table 1: Overview related works and CAvA approach. ●fully meets, ◑Partially ○does not attend.

–Evaluated approaches–	AF [8]	AA [7]	RA [6]	AS [5]	CMA [4]	AT [3]	AI [2]	MCS [1]
PORE	●	○	◑	○	○	●	◑	●
MiHOS	○	○	●	○	◑	●	●	○
UnHOS	○	●	●	●	◑	●	●	○
DesCOTS	◑	◑	◑	●	○	●	○	○
DSS	○	●	◑	●	○	●	○	○
ArcheOpterix	○	●	●	●	●	●	●	○
GATSE	○	●	◑	●	○	●	○	○
RossTool	●	●	○	○	●	●	○	○
CAvA	●	●	◑	●	●	●	●	●

The novelty of CAvA approach lies in its holistic view of CPS architecture systems, when tradeoff multiple S&A COTS devices occur, applying MDE - in the sense of Model-Based Analysis - by performing the "ability to evolve" in response to changes in requirements, assessing and analyzing a set of scenarios generated from the evaluation of multiple functionalities and hardware interfaces.

3 CAvA APPROACH

The CAvA approach supports planning, definition, and analysis of S&A COTS device tradeoff, component modifications, requirements, software quality attributes, and architectural impacts.

It consists of five phases and related criteria: planning, design of the evolved architecture (MCS, CMA, AS), analysis of the candidate architectures for evolution(AI, RA, AA, AF), architecture optimization, and evolved architecture consolidation. Each phase has a set of four activities that guide designers in the development of the evolution of architecture. Performing the activities planned in each step feeds the next step, as illustrated in Figure 1.

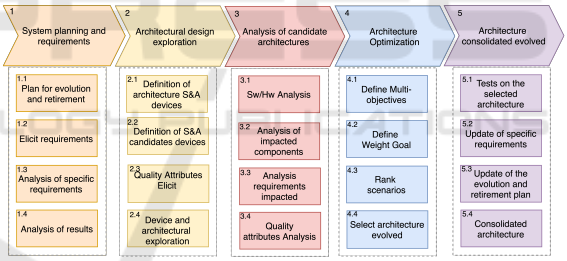

Figure 1: CAvA phases and activities.

The details of the proposed phases and activities are the following:

Phase 1 ***Planning and System Requirements*****:** *The survey of project requirements, specific requirements, system architecture operation, and results of the analyses performed on the architecture are carried out.*

Activity 1.1 ***Plan Evolution and Retirement*****:** *The stackholders, constraints, and limitations defined in the project are lifted to meet the requirements of the*

[1]MCS: Multiple COTS selection.
[2]AI: Ability to address incompatibilities.
[3]AT: Available tool.
[4]CMA: COTS modeling and architecture.
[5]AS: Approach with norms standards.
[6]RA: Requirements analysis.
[7]AA: Analysis attributes.
[8]AF: Analysis functionalities.

application. The information collected should direct designers in the survey of modified design.

The Evolution and Retirement Plan (PEA) periodically evaluates the potential improvements and inclusions of new features based on the technological advancement of S&A devices, starting the registration of the ability to make modifications and inclusion of new requirements.

Activity 1.2 ***Eliciting Requirements*****:** *The project requirements are raised, including its functionalities and details of the system in operation. Requirements with the potential to be updated, or requirements that can be included in the future, should be pointed out.*

Operational design requirements that will be modified with the tradeoff/inclusion of candidate S&A devices are identified and the specific requirements related to the devices are identified and selected. The inclusion or updating of requirements can directly impact the architecture, and a detailed analysis of the specifications of the components and composition of the system is performed in the next activity.

Activity 1.3 ***Analysis of Specific Requirements*****:** *The specific requirements, features, and characteristics of the current architecture are raised, identifying components and functionality tied to design requirements.*

The functional and non-functional requirements of the architecture in operation are analyzed identifying the relationship with the components, their characteristics, and specific properties. For cases where new requirements and functionality are included, the designer must indicate the components added to the architecture.

Activity 1.4 ***Analyze Simulation and Architecture Results*****:** *The results of the simulations, tests, and validation of the current system are obtained. System quality attribute data is used as a reference to evaluate evolution scenarios.* The results obtained in simulations, bench tests, validation and architecture analyses are used as a reference to evaluate the potential scenarios of architectural evolution. This data is used in the architectural model for further analysis of the quality attributes, deployment, and current parameters of the architecture.

Phase 2 ***Evolved Architecture Design*****:** *The S&A devices of the current architecture and candidates are surveyed to analyze functionality, hardware interface and requirements.*

Activity 2.1 ***Definition of Architecture S&A Devices*****:** *The potential S&A devices of the architecture in operation that must be replaced are identified.*

Designers receive new system requirements demands that are linked directly to the tradeoff or inclusion of new S&A devices, starting the selection and definition of s&A devices that must be replaced in the architecture.

Activity 2.2 ***Definition of S&A Devices Candidates*****:** *The potential S&A devices that are candidates for replacement or inclusion in the operating system architecture are identified.*

Designers conduct a technical market research on the availability of S&A devices with physical characteristics and properties compatible with the system architecture and that meet the new requirements of the project. After listing candidate devices, the designer performs the modeling for further compatibility analysis.

Activity 2.3 ***Eliciting Quality Attributes*****:** *The software and system quality attributes are elicited. The physical, electrical, hardware interface, and software components characteristics and properties are identified.*

The quality attributes are structured based on the ISO/IEC 25010(ISO/IEC, 2011) software quality standard, consisting of 8 characteristics: functional adequacy, reliability, performance efficiency, compatibility, usability, safety, maintainability and portability. Each characteristic has a set of sub-characteristics and quality attributes that can be declared in the scope of the architectural model for analysis of candidates for further attributes valuation.

Activity 2.4 ***Architectural Exploration*****:** *Scenarios are exploited based on functionalities and attributes, evaluating the hardware interface between the system and the candidate devices.*

The exploration of architectures takes place in tree tasks. In the first task there is the evaluation of COTS candidates functionalities compatible with the old devices and interface available in the architecture and embedded platform. In the second task, a set of scenarios are generated that cover the selected functionalities and interface connections to the architecture. Multiple COTS functionalities are combined with candidates generating a set of scenarios. The third task are Feature-friendly scenarios that have no interface with the architecture are created with the addition of converter devices and connections.

Phase 3 ***Analysis of the Candidate Architectures for Evolution*****:** *In this step the designer analyzes the generated scenarios, quality attributes, evaluating compliance with requirements and impact of the tradeoff.*

Activity 3.1 ***Analysis of Software and Hardware*****:** *The designer performs the evaluation of the software and hardware components of the generated scenarios and selected architecture.*

The scenarios generated are analyzed by identifying hardware and software components, such as: ports, communication and connection bus, tasks, processes, and data type. S&A devices that internally

have a dedicated embedded system to perform specific signal handling functionalities, can encompass architecture software components that until then were run on the embedded system platform. With this, redistribution, removal or adjustments in processes and tasks can add integration elements such as wrappers and code glue that are analyzed in the next activity.

Activity 3.2 ***Analysis of Impacted Components*****:** *Impact analyses are performed between scenarios and selected architecture.* The scenarios generated are compared with the selected architecture identifying the hardware and software components that were impacted by the tradeoff. The survey of hardware components, such as: interface, connections, power and bus; software components, such as: connections, data types, communication bus speed configuration, processes, and threads that have had modifications to suit the integration of the new device, being compared with the selected architecture, identifying the differences and providing information extracted from the scenarios.

Activity 3.3 ***Analysis of Impacted Requirements*****:** *The survey and analysis of system requirements that have been impacted by the exchange of S&A devices is performed.*

The analysis of the functional and non-functional requirements that have been impacted by the change of the S&A device occurs. The designer performs the survey of attributes by type of characteristic, relating to the defined requirements and performs the analysis to identify potential risks in the scenarios explored.

Activity 3.4 ***Analysis of Scene Quality Attributes*****:** *In this activity analyses of the quality attributes of the scenarios are performed.*

The result of the analyses obtained in activities 3.1 to 3.3 and the evaluation of specific attributes linked to S&A devices, such as: accuracy, resolution, signal ratio, noise, sensitivity, hysteresis, output torque, temperature operation are obtained to make up the set of software and system quality attributes, being structured according to the definitions performed in activity 2.3.

Phase 4 ***Architecture Optimization*****:** *The quality attributes of quantifiable scenarios are processed to provide a set of parameters for each identified characteristic type and attribute.*

The attributes of the extracted scenarios and parameters of the quantifiable design requirements are used in decision-making techniques such as Analytical Hierarchy Process (AHP). To properly set the analysis, each attribute is defined with a minimum or maximum value. For example, the attribute price is defined with minimum value and processor capacity with maximum value.

Activity 4.1 ***Definition Multi-objectives*****:** *The designer defines which objectives are wanted to be evaluated based on the quality attributes elicited.*

In this activity there is a definition of which characteristics, subcharacteristics and quality attributes the designer wants to evaluate in the scenarios. Each scenario provides information about quantity of components, device converters added, wrappers and connections are impacted when compared with architecture selected. In this way, the designer can also define objectives considering the impact of the modifications necessary to choose the scenario.

Activity 4.2 ***Definition of Weight Goal*****:** *The designer defines the weights of each attribute based on objectives he wants to evaluate.*

According to the objectives defined for the exploration of the scenarios, the designer defines weights to the attributes to rank the scenarios.

Activity 4.3 ***Rank Scenarios*****:** *The ranking of scenarios is performed according to the objectives and weights defined.*

The ranking of the scenarios is performed by defining which scenarios have achieved the best results and have met the requirements.

Activity 4.4 ***Select Architecture Evolved*****:** *The designer selects the evolved architecture he wants to implement.*

Phase 5 ***Consolidate Evolved Architecture*****:** *In this step tests and validation of the architecture are performed with the new defined parameters. Requirements update and PEA are performed.*

Activity 5.1 ***Tests and Validation*****:** *In this activity the tests and validation of the evolved architecture are performed.*

In this activity, the use of techniques like Hardware in-the-loop (HIL), Software in-the-loop (SIL), and Processor in-the-loop (PIL) are suggested to extract behavior and performance data from the evolved architecture in a real simulated environment. The test results are used to feed, recursively, **Phases 2 and 3**, refining the architectural model and performing new analysis.

Activity 5.2 ***Project Requirements Update*****:** textitIn this activity, the specific design requirements obtained from the architecture performance data and tests performed are updated.

The new design requirements obtained from the performance analysis of the evolved architecture with the definition of the new deployment are validated. The modifications and inclusion of new features that culminated in the fulfillment of the new design requirements are updated in the PEA for further evaluation of potential improvements in the system.

Activity 5.3 ***Evolution and Retirement Plan Update*****:** *The evolution plan and PAE update take place, and all changes made to the evolved architecture are documented.*

All exchange records of S&A devices are collected and updating project documents and requirements to be used as a reference.

Activity 5.4 ***Consolidated Architecture*****:** *The designer upgrades the architecture to the new selected architecture. New evolution's/modifications are based on the consolidated architecture.*

The consolidation of the evolved architecture occurs after the other proposed activities have been carried out. The evolved architecture is defined as the current system architecture, and its upgraded version is indicated. New developments and modifications can be made based on the consolidated architecture.

A tool named *Automatic Evolution Wizard (AEW)* was developed to support the application of CAvA for automatically explore and analyze possible scenarios, specifically from phases 2 to 4. Phases 1 and 5 are performed manually as they need multiple tools for testing and validation that are not yet integrated.

4 CASE STUDY

To illustrate the architectural evolution approach proposed in this paper it was applied to modify(evolute) the architecture of an (UAV) projects. These projects are part of a research effort called Provant, initiated in 2012 in partnership between UFSC and UFMG. Since the beginning of ProVant, Four aircraft prototypes have been developed, as shown in Figure 2. Each successor version aims enhance the problems encountered in the predecessor project and also to add new functionality.

Figure 2: ProVant prototypes versions.

Applying the proposed approach, the planning step **Step 1**) begins with the survey of project requirements, defined specific requirements based on the application and characteristics of the UAV of the current version. Resulting in the architectural and requirements models, as well as the previous analysis performed.

To exemplify, part of requirement model specification of UAV 3.0 with the *ReqSpec* language file is shown in the Listing 4. The requirements: UAV weight limit (lines 3-8); inlet power(lines 9-14).

Listing 1: Requirements model in ReqSpec language.

```
1  system requirements reqs for ProVant_3_0 [
2  description "These are requirement for UAV"
3  requirement R1: "UAV weight limit"[
4   val  MaximumWeight = 1.2 kg
5   category Quality.Mass
6   description this
7  "shall be within weight of" MaximumWeight
8   value predicate MaxWeight==#SEI::WeightLimit
9  see goal SCFgoals.ngl]
10 requirement R2:"UAV inlet power" for UAVSystem[
11  val MaximumPowerBudget = 5.0 W
12  category Quality.Performance
13  compute actualvolt:Physical::Voltage_Type
14  value predicate MaxPowerBudget==PowerBudget
15 see goal SCFgoals.g2]
```

Based on the *ReqSpec* model, the architecture candidate's requirements are analyzed.

The UAV 3.0 architecture model is presented in Listing 2. It is composed by three processors: STM32F405, nrf51822 and and BEAGLEBONE (ARM Cortex-A8). The software processes are: RF_Firmware, STM32F405_Firmware, and BEAGLE_Firmware (lines 6-8). It includes devices and buses (lines 9,10), connections (lines 11-13), and resources/binding allocation (lines 15-17). As an example, it has been selected to tradeoff the GY85 COTS device, an IMU with three sensors: ADXL345 (acceleration), ITG3205 (gyroscope), and HMC5883L (magnetic field).

Listing 2: AADL model selected to evolve.

```
1  system implementation ProVant_3_0.impl
2   subcomponents
3    STM32F405:processor STM32F405;
4    nrf51822:processor nrf51822;
5    BEAGLEBONE:processor STM32F405;
6    RF_Firmware:process nRF51822_Firmware;
7    STM32F405_Firmware:process STM32F405_Firmware;
8    BEAGLE_Firmware:process Beagle_Firmware.impl;
9    GY85:device GY85;
10   ... (10 architectural components)
11  connections
12   C0: bus access STM32F405.uart_bus -> UART;
13  ---- (17 connections)
14  properties
15     Actual_Processor_Binding => (reference
16     (STM32F405)) applies to STM32F405_Firmware;
17          ... — (12 Allocations and bindings)
18 end ProVant_3_0.impl;  \label{aadlcode}
```

The COTS candidates that were surveyed must have the same functionalities and be compatible in terms of properties when compared with the selected device. Candidates that do not have the functionalities to be replaced totality can be combined with other candidates.

In this example, the selected GY85 device has three functionalities (accelerometer, gyroscope, magnetometer). Four devices surveyed can potentially cover the functionalities and hardware interface compatibility. Another 8 devices cover partially the functionalities, using more than one combined device. In total, 12 candidate devices were elicited for aadl modeling (CAvA library) to explore the scenarios, as shown in Table 2.

Table 2: Candidate devices elicited for tradeoff.

Candidates	Accel.	Gir.	Comp.	Magnet.	Baro.	Temp.
GY85 (selected)	O	O	O			
BN055	X	X	X			
ICM20948	X	X	X			
ADIS16480	X	X	X	X	X	
MPU9250	X	X	X			
GY955	X	X		X		
ICM20649	X	X				X
ADIS165052	X	X				X
LSM303DLHC	X			X		
ADXL345	X					
HMC6352			X			
LSM303	X			X		
L3GD20H		X				

A video[1] is made available to show the AEW tool usage within the present study. In **Activities 2.1 and 2.2** the designer selects the AADL and ReqSpec models, and defines the implemented system to be evolved. The tool reads the models, analyzes the functionality of the devices selected for replacement, and search for candidate devices. Next, in **Activities 2.3 and 2.4**, it presents the list of devices to be replaced and create possible tradeoff scenarios.

Activities 2.3 and **2.4** are performed generating the set of scenarios explored in tree sequential tasks, based in functionalities and quality attributes from hardware interface characteristics. The first task generates scenarios with candidates and combinations of candidates that fulfill selected functionalities. The second stage generates scenarios with hardware interfaces (ports and bus) compatibles. Candidates with multiple hardware interfaces compatible generate alternatives of scenarios. The third stage generates scenarios without hardware interfaces compatible, adding a converter device in hardware interface for integration with the architecture. The focus is to consider as many scenarios as possible to be analyzed and further rank them.

Considering the selection of devices that have the same hardware interface as the GY85 and functionalities, the approaches described in the related works would be able to also solve this challenge due their specific focus on properties. However, evaluating the hardware interface candidates, only 4 devices within the 12 elicited are enabled, reducing the exploration of scenarios and limiting the space project and multiple objectives.

In total, applying CAvA approach, 126 scenarios were generated containing the comparative analysis with the selected architecture. The CAvA approach expands the generation of scenarios based on evaluation of functionalities, compatible and non-compatible hardware interfaces.

The AEW tool presents the created scenarios and changes details to integrate the candidate e combinations. In the Listing 4 shows the changes from scenario with MPU9250 device tradeoff, declaration added to integrate the device (lines 1-4), changes in connections (lines 5-7), device that is deleted from the architecture (lines 8,9), functional suitability from GY85 to MPU9250 (lines 10,11), and information to add a wrapper in C11 connection to provide compatibility (lines 12,13).

Listing 3: AADL model selected to evolve.

```
1  declaration added:
2    subcomponent MPU9250
3    subcomponent wrapper_DOF6_to_DOF5
4    connection C23
5  declaration changed:
6        connection C11
7        connection C4
8  declaration deleted:
9    subcomponent GY85
10 Functional Suitability:
11  MPU9250.impl: GYROSCOPE,ACCELEROMETER,COMPASS
12 Messages:
13 Wrapper was used in C11:MPU9250.DOF6->STM32F405.DOF5
```

Analyzes of candidate scenarios, shown in **Phase 3**, are processed identifying modifications and impacts on architecture components, quality attributes and requirements. Therewith, the results compose the quality attributes that are analyzed and structured according to characteristic and sub-characteristics, described in **Activity 3.4**, to support the architecture optimization.

And finally, the architecture optimization is contemplated in AEW tool. The attributes and result of scenarios analysis are presented following quality characteristic and normalization of ratings, from 0 to 1, adjusting values measured on different scales, where the designer defines the weights for each desired objective, as defined in **Activities 4.1 and 4.2**. In total, 22 attributes are modeled. After the set definitions, the tool generates the ranking of scenarios, **Activity 4.3**. The designer can select the scenario he wants to implement and generate the AADL model of the evolved architecture, **Activity 4.4**.

The UAV 3.0 attributes and result of scenarios analysis as number of connections, devices, modifications, features and properties are normalized, from 0 to 1, and are presented to the designer who further performs the decision-making to calculate the ranking of scenarios. For the present study, the objectives and related weights defined to architecture evolution are better precision, lower noise signal, of the data provided from gyroscope(x0.3) and accelerometer(x0.3), lower hardware conversor(0.2) and software wrapper(0.2) added as shown in Figure 3. In total, 34 scenarios demonstrate better results than original architecture. The best obtained scenario was with the ADIS16480 device integrating multiple functionality with the least impact of change in architecture. How-

[1] https://youtu.be/v1kZBcd4yVM

ever, this scenario presented highest cost, increased area, power consumption, and also increased data consumption on the SPI bus compared to GY85 device presented in the tool's analysis results.

characteristic	attribute	range min	range max	reference	score
functionality	Integrated Software	0	0		0
functionality	Software Wrapper	0	3		0.20
functionality	Hardware Conversor	0	2		0.20
maintainability	Subcomponents Total	17	23		0
maintainability	Connections total	23	31		0.60
performance	Weight Total	0.2880000...	0.392000...	9 Kg	0.00
performance	Price Total	0 Dollar	817.5 Dollar	8	0
performance	BEAGLEBONE Usage ...	0.0120000...	0.012000...	300 MIPS	0
performance	BEAGLEBONE Usage ...	0.0299999...	0.029999...	300 MIPS	0
performance	nrf51822 Usage Min	0 MIPS	0 MIPS	30 MIPS	0
performance	nrf51822 Usage Max	0 MIPS	0 MIPS	30 MIPS	0
performance	STM32F405 Usage ...	0.0130000...	0.013000...	150 MIPS	0

Figure 3: Weight definition to scenarios ranking.

The result of the analysis is made available in a .csv file where the designer evaluates the details of changes, quality attributes with related goals, and analyses results from selected scenario. After testing and validation of the architectural model, **Activity 5.1**, the designer updates the design, evolution plan and retirement requirements described in **Activities 5.2 to 5.4**. The architectural model is consolidated and represents the current architecture, as described in **Activity 5.4**, the results of the analyses, tests, adjustments, and new requirements are updated.

5 CONCLUSIONS AND FUTURE WORK

CavA suggests a set of phases and activities that, in cooperation with additional model analysis tools, help to mitigate these risks and limitations, automatically evaluating the impacts of modifications using quality attributes declared in the AADL model. Comparing with the related works, CAvA expands the capacity to evaluate scenarios based on the functionality, compatible and non-compatible hardware interface, enabling the exploration of emerging technologies.

A difficulty in tracking functionalities suitability between requirements defined in ReqSpec and AADL models was identified. ReqSpec does not support the declaration of functionalities, demonstrating a gap to support functionality that can be further investigated.

REFERENCES

Aleti, A., Bjornander, S., Grunske, L., and Meedeniya, I. (2009). Archeopterix: An extendable tool for architecture optimization of aadl models. In 2009 ICSE Workshop on Model-Based Methodologies for Pervasive and Embedded Software, pages 61–71.

Barbacci, M., Carriere, S., Longstaff, T., Weinstock, C., and Feiler, P. (1998). Steps in an architecture tradeoff analysis method: Quality attribute models and analysis. Technical report, CMUSEI.

Bass, L., Clements, P., Kazman, R., et al. (2013). Software architecture in practice.

Bengtsson, P., Lassing, N., Bosch, J., and van Vliet, H. (2004). Architecture-level modifiability analysis (alma). Journal of Systems and Software, 69(1-2):129–147.

Feiler, P. H., Delange, J., and Wrage, L. (2016). A requirement specification language for aadl. Technical report, CMU/SEI.

Garg, R. (2017). A systematic review of cots evaluation and selection approaches. Accounting, 3(4):227–236.

Grau, G., Carvallo, J. P., Franch, X., Quer, C., et al. (2004). Descots: a software system for selecting cots components. In Proc.. 30th Euromicro Conf., 2004., pages 118–126. IEEE.

Ibrahim, H., Elamy, et al. (2011). Unhos: A method for uncertainty handling in commercial off-the-shelf (cots) selection. Int. Journal of Energy, Information & Communications, 2(3).

ISO/IEC (2011). Iso/iec 25010: 2011 systems and software engineering–systems and software quality requirements and evaluation (square)–system and software quality models.

Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., and Carriere, J. (1998). The architecture tradeoff analysis method. In Proc. 4th IEEE Int. Conf. on Engineering of Complex Computer Systems, pages 68–78. IEEE.

Lee, E. A. and Seshia, S. A. (2016). Introduction to embedded systems: A cyber-physical systems approach. MIT Press.

Mohamed, A., Ruhe, G., and Eberlein, A. (2007). Cots selection: Past, present, and future. In 14th IEEE Int. Conf. and Works. on the Eng. of Computer-Based Systems (ECBS'07), pages 103–114.

Mohamed, A., Ruhe, G., Eberlein, A., et al. (2007). Mihos: an approach to support handling the mismatches between system requirements and cots products. Requirements Engineering, 12(3):127–143.

Ncube, C. and Maiden, N. A. (1999). Pore: Procurement-oriented requirements engineering method. In Int. workshop on component-based software engineering, pages 130–140.

Procter, S. and Wrage, L. (2019). Guided architecture trade space exploration: Fusing model based engineering design by shopping. In 2019 ACM/IEEE 22nd Int. Conf. on Model Driven Eng. Languages and Systems, pages 117–127.

Ross, J. A., Murashkin, A., Liang, J. H., Antkiewicz, M., and Czarnecki, K. (2019). Synthesis and exploration of multi-level, multi-perspective architectures of automotive embedded systems. Software & Systems Modeling, 18(1):739–767.