Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

AdaptiveSystems: an integrated framework for adaptive systems design and development using MPS JetBrains domain-specific modelling environment

Sofia Meacham[1], Vaclav Pech[2], and Detlef Nauck[3]

[1]Department of Computing & Informatics, Bournemouth University, Poole, BH12 5BB, UK
[2] JetBrains MPS, Prague 140 00, Czech Republic
[3] British Telecom, Ipswich IP5 3RE, U.K.

Corresponding author: Sofia Meacham (e-mail: smeacham@bournemouth.ac.uk).

This work was financially supported by research funding allocated directly from British Telecom (Data Science team in BT Research Headquarters in Adastral Park) to Bournemouth University.

ABSTRACT This paper contains the design and development of an adaptive systems (*AdaptiveSystems* Domain-Specific Language - DSL) framework to assist language developers and data scientists in their attempt to apply Artificial Intelligence (AI) algorithms in several application domains. Big-data processing and AI algorithms are at the heart of autonomics research groups among industry and academia. Major advances in the field have traditionally focused on algorithmic research and increasing the performance of the developed algorithms. However, it has been recently recognized by the AI community that the applicability of these algorithms and their consideration in context is of paramount importance for their adoption. Current approaches to address AI in context lie in two areas: adaptive systems research that mainly focuses on implementing adaptivity mechanisms (technical perspective) and AI in context research that focuses on business aspects (business perspective). There is currently no approach that combines all aspects required from business considerations to appropriate level of abstraction. In this paper, we attempt to address the problem of designing adaptive systems and therefore providing AI in context by utilising DSL technology. We propose a new DSL (*AdaptiveSystems*) and a methodology to apply this to the creation of a DSL for specific application domains such as *AdaptiveVLE (Adaptive Virtual Learning Environment)* DSL. The language developer will be able to instantiate the *AdaptiveSystems* DSL to any application domain by using the guidelines in this paper with an integrated path from design to implementation. The domain expert will then be able to use the developed DSL (e.g. *AdaptiveVLE* DSL) to design and develop their application. Future work will include extension and experimentation of the applicability of this work to more application domains within British Telecom (BT) and other areas such as health care, finance, etc.

INDEX TERMS language composition, reusability, DSL, adaptive systems

I. INTRODUCTION

There is no doubt that the software and systems of the future will be dominated increasingly by adaptive systems. The static cases that were used for many years by several application domains from legacy software to embedded systems will have to move to more dynamic models. These dynamic models range from simpler adaptive models where information is collected and changes are made with human intervention (human-in-the-loop systems) to more complex models where information is collected from the environment at runtime and the system is able to adapt itself (self-adaptive systems) [1].

There is a lot of controversy in the literature for the terms adaptive, self-adaptive, autonomic [2], intelligent and smart systems. Typically, the term smart or intelligent systems is used for Internet of Things (IoT) applications that contain physical devices to collect information such as sensors [3] whereas the rest of the terms have not been associated with such devices. For simplicity and clarity, we will use the term adaptive systems to refer to all the systems that collect and process data through any means (hardware or software sensors) with different levels of human intervention.

It has been widely recognised by the AI community and was repeatedly raised by BT's data science team through their twenty years of experience within BT research that a major issue in the design and development of adaptive systems is the lack of an overall high-level system description. This need had been identified early on in attempts to design adaptive systems such as in [4] were a high-level design oriented language for an adaptive web systems framework was developed. Moreover, the technical complexity of these systems makes them harder to adopt in different application contexts and a system-level approach is an emerging area with research interests for complex domains such as IoT [5] [6] [7]. Although there are solutions and frameworks that have been developed, they mainly focus on implementation details - for more refer to Section II. Therefore, the need to abstract at a higher-level in simple and comprehensive steps and give an overview of how the system will operate in context is an area that AI community is currently identifying as a strong requirement. On the other hand, research in providing AI in context has emerged recently – for more please refer to Section II. However, it mainly focuses on business aspects without tying them with the corresponding technical implementations. Domain-specific modelling is an emerging area that started from MDE and provides a higher-level of abstraction, isolation of domain-specific aspects with excellent tooling and tailoring to particular domain requirements that include "intelligent" systems (machine learning algorithms, adaptive systems) [8] (DSLs for big data – machine learning) [9].In our previously published work [10] we defined a framework for developing adaptive VLEs. In this paper, we raised the level of abstraction even more by defining an adaptive system language-framework that can be ported to several application domains such as the Adaptive VLE application, the education IoT application, etc. From this high-level description, we revised the previously developed adaptive VLE framework to an improved version and complemented it with corresponding data handling languages. The MPS JetBrains domain-specific languages (DSLs) development environment [11] was used to develop a new language. We mainly utilised its strong language composition characteristics in combination with the code generation features [12] to create the path between adaptive systems and specific application domains. Language extension and composition were the main elements that provided this further raising of abstraction.

It is worth noting that in this paper, we focus on the system level. Specifically, we focus on the adaptive, smart, intelligent elements that integrate the existing applications with data science algorithms. The level of autonomicity and/or human intervention provided is outside the scope of this work. There is substantial work in the literature for making systems autonomic but that is at a lower level of abstraction. Also, the term integrated framework refers to many different aspects from the tooling that integrates the design and implementation to the integration of diverse paths such AI software and traditional static software. A comprehensive comparison with other related work is presented in the background section as integration is an important part of the proposed work.

Concluding, the main steps of our proposed work are summarized as follows: an adaptive systems language that will enable AI in context was developed to address to industrial needs and cover existing research gaps in this area such as integrated path from design to implementation and tooling. This language followed MAPE-k loop architectural elements (block diagram descriptions with detailed explanations are provided in Figures 1, 2, 4) and had main purpose to assist and provide a methodology for language developers of many application domains. The range of application domains that this work can be applied will potentially create tremendous positive impact in the field of adaptavie systems. The adaptive VLE application domain was used as proof of concept and guidelines for future application to other domains were provided. Last but not least, we evaluated the developed DSL through proper language quality characteristics evaluation method with feedback from the industrial users (BT) and language designers (MPS JetBrains).

A background study on adaptive systems and AI in context will create the requirements for this research and is presented in Section II. In Section III, our proposed solution through the development of an integrated framework will be given in detail. A case study was used to demonstrate the proposed language design and development process: the adaptive VLE. All the DSLs developed will be described with two main subsections each: language structure and language usage. The emphasis will be in their reusability and composition. Five main languages were used: *AdaptiveSystems* (new DSL), *Adaptive_VLE* (new revised DSL) for collecting data, *datamapping* (previous publication) [13] for defining the features and their type, *datamapping_Adaptive_VLE* (new revised DSL) for connecting the *datamapping* with the *Adaptive_VLE* application domain language, *Classification Algorithms Framework (CAF)* (previous publication) [13] language for data processing using a range of classification algorithms. In Section IV, the main benefits from the proposed solution will be detailed with a separate subsection on the applicability to other application domains. In Section V, BT's and MPS's language engineers' evaluation will be presented. Finally, our conclusions and future plans will be detailed in Section VI.

II. BACKGROUND STUDY - ADAPTIVE SYSTEMS AND AI IN CONTEXT

A. Adaptive Systems

Adaptive systems have been part of AI research for several years now. Although there is controversy surrounding the definition of the terms adaptive, autonomic, self-adaptive, smart, intelligent the common denominator is that systems of the future will not be static anymore [14]. They will have to change both in structure and behavior to adapt to changing requirements, environment conditions.

The need to support the design and development of adaptive systems has led to the application of several software engineering techniques [15]. Specifically, architectural description languages, adaptation frameworks and aspect-oriented techniques have been used to achieve adaptivity. All these techniques were used to incorporate adaptation mechanisms to enable change during runtime.

The most known approaches that have been published in this research field are summarized as follows: The Mobility and Adaptability enabling Middleware (MADAM) [16] project provides a general component model and middleware infrastructure that supports various adaptation styles for mobile applications. Sadjadi *et al.* developed the Adaptive CORBA Template (ACT) to enable runtime improvements to CORBA applications in response to changing requirements and environmental conditions [17]. ACT transparently weaves adaptive code into an object request broker (ORB) at run time. The woven code intercepts and modifies the requests, replies, and exceptions that pass through the ORBs. One of the advantages of ACT is that it is language and ORB independent. A dedicated language called Stitch has been presented in [18] and is used as part of the Rainbow framework. The Rainbow framework is the most well-known adaptivity framework and has been applied to many application domains. Additionally, tools such as SOTA [19], ACTRESS [20] and CYPHER [21] have been presented in the literature. The DSML which is domain specific modelling language for Dynamic Adaptive Systems is another example of work that focuses on the adaptivity mechanisms [22]. Similarly in [23][24], the emphasis is on modelling the adaptivity mechanisms and their verification.

However, all the above approaches focus on how to enable adaptivity at a very near to implementation level by focusing on implementing adaptation mechanisms. A higher-level of abstraction that will enable to consider the elements of adaptive systems in context is still missing.

Specifically, all the above approaches either address specific types of applications such as mobile applications making them not easy to be reused and/or they lack tool support for separating implementation and technical details from the application domain. This would abstract the development process and make it accessible to non-technical users (users with domain expertise other than software development for example such as health, education professionals, etc.).

However, there are some approaches that overcome the above-mentioned problems and are nearer to our proposed solution.

The most remarkable work comes from Arcaini et. al in [25]. The authors developed MSL (textual DSL) which stands for MAPE Specification Language and it implements MAPE-K patterns defined by Weyns in [26] for decentralized self-adaptive systems. It contains formal analysis and a case study for smart-home systems development. However, this work contains a quite abstract description of MAPE-K and would require the interfacing with other tools such as verification, analysis and running platform tools. This would include model to model transformations which is a difficult process in most cases therefore making the path from design to implementation segmented. In our work, we propose a high-level system architecture description with encapsulated path to implementation through MPS's model to model transformations. We follow the "master-slave" pattern as it is the most used in the industry and our approach covers a representative range of adaptive systems applications as was evidenced by BT's researchers that participated in this work.

Another work worth mentioning is the MiDAS model-based environment for adaptive systems proposed by José Bocanegra in [27]. This work utilises a graphical DSL representation that would have difficulties scaling as is recognised for all graphical representations [28]. It is tailored to the case study developed and its applicability to other application domains is unclear, mainly referring to adaptive interfaces.

B. AI in Context

Current literature for enabling AI in context mainly comes from the business perspective. Mikael Berndtsson in two recent publications details the nontechnical factors that should be considered to enable AI adoption. In [29], the need for nontechnical strategy for scaling analytics in organisations by proposing an initial plan is presented, whereas in [30] information gathered from 13 organisations confirmed the value of nontechnical strategies and suggested the application of change management techniques [3]. In these papers, the nontechnical aspects are addressed through techniques from business related research anticipating that AI has similar requirements with other Information Systems (IS) as is also stated in [31]. In [31] an approach of modelling the AI in context using formalism from the business processes modelling research such as Archimate [32] is being proposed. This approach is closer to the system architecture although still focuses on business aspects. Other research in the area of AI in context focuses on technical perspectives such as algorithmic improvements to provide AI explainability in context [33][34].

Both the above research areas approach the problem from different perspectives and solve only part of it. The adaptive systems part focuses on technical details for implementing adaptivity and rarely considers its adoption by application domains. The AI in context part, focuses on the business and nontechnical aspects and doesn't include the peculiarities that make AI harder than the usual ISs.

In this paper, we will attempt to bridge the gap between the approaches by creating a new domain-specific language (DSL) to address the problem. This new DSL will hide the implementation details from the nontechnical users, address the most common aspects that adaptive systems design entail (AI technical requirements), abstract the complexity of these systems, provide multiple views for all domain users involved. Although it uses domain-specific technology to solve the specific problem, our solution is not specific to an application domain and can be applied easily to other domains without the need to understand the details. This seems like a controversy as we use a domain-specific technology to create a domain-independent framework. However, it is not the first time that domain-specific languages have been used in this way [25]. The requirement to address the research gap of providing AI in context contains the build of a framework that is domain-independent or else stated application independent.

Specifically, we will base our work on creating a new language and toolset for the design and development of adaptive systems mainly based on the MAPE-K loop model using a centralized control, architecture-based approach targeting different application domains for adaptive systems. The reasons for these choices are that they are currently used extensively in industrial applications. To demonstrate the applicability of the proposed method, we will use a case study: The Adaptive Virtual Learning Environment (AdaptiveVLE) (IS).

III. PROPOSED SOLUTION – *AdaptiveSystems* - INTEGRATED FRAMEWORK FOR ADAPTIVE SYSTEMS

A. *CASE STUDY – ADAPTIVE VLE*

The AdaptiveVLE case study consists of the design and development of a fully online VLE that uses adaptivity to provide personalisation and therefore enhances the online education provision.

Specifically, the educator will configure what data will be collected by the AdaptiveVLE, then use the AdaptiveVLE in the online class and collect learning analytics data to personalise each student's learning path. For example, students with high academic achievements will be given more advanced resources whereas students with low achievement will be given more resources for basic skills.

This falls under the general category of adaptive systems as it collects, analyses, and processes data and acts on the results from the data processing. Specifically, it corresponds to adaptive web systems as the underlying structure of the VLE is a web development software for the VLE.

B. *PROPOSED ARCHITECTURE - MAPE-K LOOP*

In Fig. 1, the MAPE-K loop is presented as was originally defined by IBM [35]. The IBM blueprint was initiated to assist the operation and maintenance of complex Information Technology (IT) systems that require highly skilled IT professionals to install, configure, operate, tune, and maintain them. To relieve the constant need of human intervention, autonomic computing was initiated. This is achieved by following the main elements of the MAPE-K loop: Monitor, Analyse, Plan, Execute and Knowledge base.

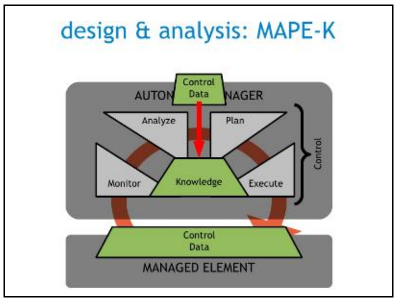


FIGURE 1. **IBM's blueprint MAPE-K architecture**

C. *DSL LANGUAGES*

Detailed description of the structure and the usage of composed languages (*AdaptiveSystems*, *Adaptive_VLE*, *datamapping*, *datamapping_Adaptive_VLE*, *CAF*) with emphasis on their composability characteristics is presented in the following subsections.

AdaptiveSystems DSL

Language structure

The *AdaptiveSystems* DSL is referring to the steps of the MAPE-K loop with a one-to-one correspondence.

Specifically, in Fig. 2, the main structure of the language is presented in a diagrammatic form. Each of the blocks correspond to concepts in MPS JetBrains.

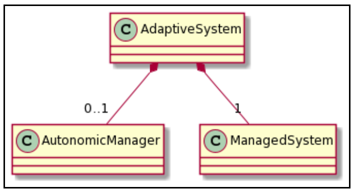


FIGURE 2. **IBM's blueprint MAPE-K architecture and its correspondence to the *AdaptiveSystems* DSL**

From the figure, we can see that *AdaptiveSystem* consists of 1 *ManagedSystem* (corresponds to MANAGED ELEMENT of MAPE-K) and 0 or 1 *AutonomicManager* (corresponds to AUTONOMIC MANAGER of MAPE-K).

The *ManagedSystem* is the static part of the system and the *AutonomicManager* is the adaptivity part that handles all data collection, processing, and adaptations. The distinction between the two parts is one of the most vital parts of the MAPE-K and along the lines that there is a strong need to separate the adaptivity functionality from the main-static system functionality [36]. The adaptive system research community has recognised that if adaptivity functionality is interleaved with main functionality then future maintenance and experimentation with different adaptation mechanisms become extremely hard to impossible.

The above MAPE-K separation between the *ManagedSystem* and the *AutonomicManager* is reinforced further and reflected by the *AdaptiveSystems* DSL structure in the MPS environment. Different folders were created to designate the separation between them. This is represented in Fig. 3.

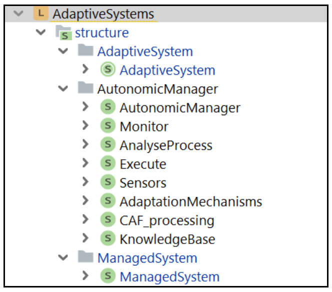


FIGURE 3. ***AdaptiveSystems* DSL organization for separation between *AutonomicManager* and *ManagedSystem***

By providing the folder structure in the *AdaptiveSystems* DSL, we enforce this separation in all the subsequent inheriting languages as will be described in the following sections.

There are more structural elements inside the *AutonomicManager* as presented in Fig. 4 (and is also depicted in part of Fig. 3).

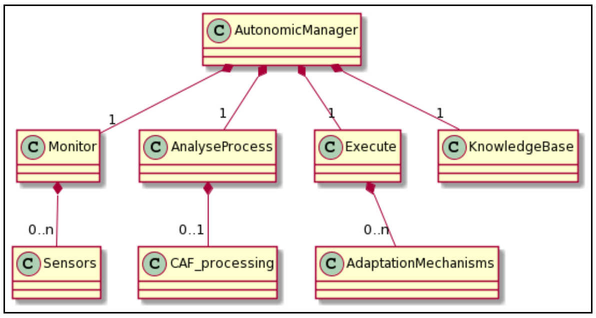


FIGURE 4. ***AutonomicManager* concept structure**

In Fig. 4, the direct correspondence between the steps of the autonomic manager of the MAPE-K loop and MPS concepts is presented. The Monitor (*Monitor* concept) Analyse Process (*AnalyseProcess* concept) Execute (*Execute* concept) – Knowledge (*KnowledgeBase* concept). The *Monitor* concept consists of *Sensors* that can be hardware or software sensors. The *Execute* concept consists of potentially many *AdaptationMechanisms*. The integration with CAF is presented in a following section.

Language usage

The *AdaptiveSystems* DSL is being extended by the *Adaptive_VLE* DSL so the usage of the language is described in the "inheriting" languages corresponding sections.

Adaptive_VLE DSL

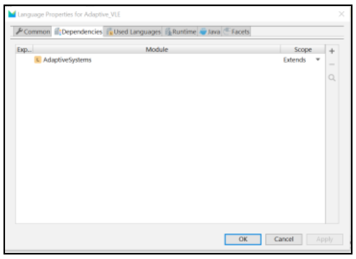
Language structure

In our previous work [37], we had developed a first version of an *AdaptiveVLE* language. In this current publication, we have extended and modified that initial version to comply with the general *AdaptiveSystems* DSL.

First of all, the *Adaptive_VLE* language is defined as an extension to the *AdaptiveSystems* language as is depicted in Fig. 5. In MPS, one language extending another is used in the same way as inheritance in object-oriented programming. Consequently, the *Adaptive_VLE* language will inherit all the concepts and their corresponding editors from the *AdaptiveSystems*. Once you declare that a low-level of detail (tailored to an application domain) DSL is extending another high-level DSL, all concepts of the high-level DSL can used and extended by the low-level DSL.

Specifically, to create the *Adaptive_VLE* DSL starting from the *AdaptiveSystems* DSL, the following steps were followed:

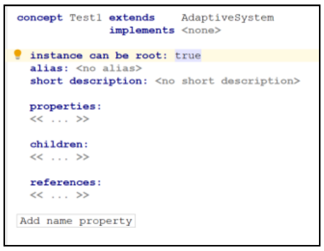
1. You declare in MPS that the *Adaptive_VLE* DSL is an extension of the *AdaptiveSystems* DSL. That is depicted in Fig. 5.



Exp...	Module	Scope
	AdaptiveSystems	Extends

FIGURE 5. ***Adaptive_VLE* DSL as an extension to the *AdaptiveSystems* DSL**

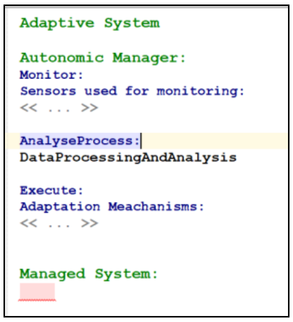
2. For concepts in the *Adaptive_VLE* DSL such as the main *AdaptiveVLE* concept, we used the "extends" relationship in the definition of a concept that refers to the concept of the high-level language. This is depicted in Fig. 6.



```
concept Test1 extends     AdaptiveSystem
              implements <none>

instance can be root: true
alias: <no alias>
short description: <no short description>

properties:
<< ... >>

children:
<< ... >>

references:
<< ... >>

Add name property
```

FIGURE 6. ***Adaptive_VLE* DSL as an extension to the *AdaptiveSystems* DSL**

The consequence of this definition is that the *AdaptiveVLE* concept inherits all the concept structure from the *AdaptiveSystems* language. Therefore, the usage of the *Adaptive_VLE* language should demonstrate that all the main elements of the *AdaptiveSystems* are inherited down. This is depicted in Fig. 7. Note that usages of the language are represented in MPS using the Solutions sandboxes.



```
Adaptive System

Autonomic Manager:
Monitor:
Sensors used for monitoring:
<< ... >>

AnalyseProcess:
DataProcessingAndAnalysis

Execute:
Adaptation Meachanisms:
<< ... >>


Managed System:
```

FIGURE 7. Solution for *Adaptive_VLE* DSL that inherits the structure from *AdaptiveSystems* DSL

3. The *Adaptive_VLE* language developer will have to develop the concepts and editors that are specific to the application domain of the low-level language.

For example, the following structures had to be defined for the *Adaptive_VLE* that don't exist in the *AdaptiveSystems* DSLs and are depicted in Fig. 8.

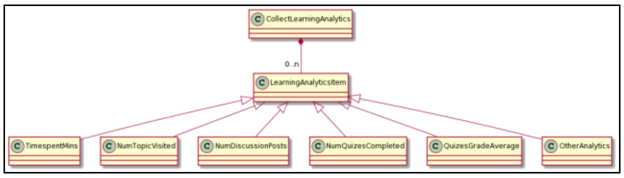


FIGURE 8. Domain-specific structure for *Adaptive_VLE* that is not inherited by *AdaptiveSystems*

Also, the *ManagedSystem* that is inherited by the *AdaptiveSystems* language needs to be extended with the main system functional concepts. For example, in the case of the *Adaptive_VLE* case study, the concepts that comprise a *BlendedCourse* should be defined under the *ManagedSystem* folder. The structure of the concepts is depicted in Fig. 9.

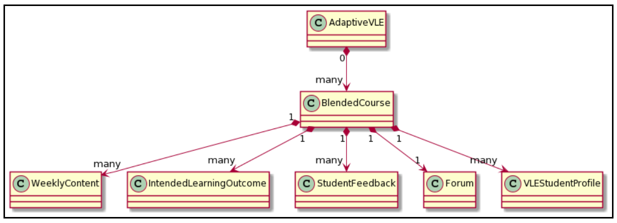
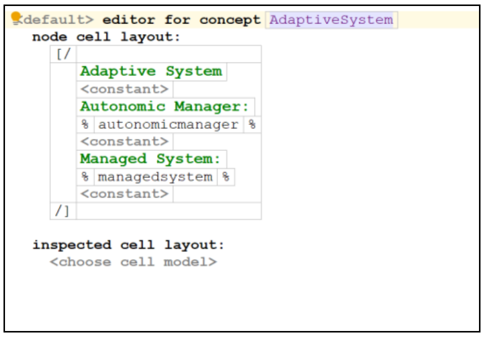


FIGURE 9. *ManagedSystem* of the *Adaptive_VLE DSL*

Language usage

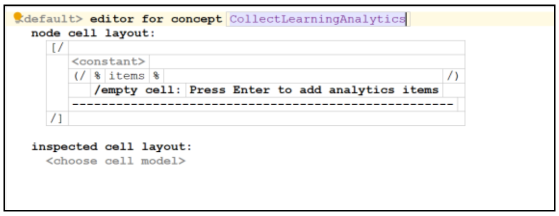
In this subsection, we 'll present the solution of the *Adaptive_VLE* DSL. Please, note that in MPS, "solution" is the usage of the developed DSL which is based on the Abstract Syntax Tree that was defined.

The developed *Adaptive_VLE* DSL uses a combination of the inherited *AdaptiveSystems* language editors and the new editors that were developed for the specific to the AdaptiveVLE domain concepts. For example, in Fig. 10, the editors for the main concepts of *AdaptiveSystems* are defined and applied in the solution figure. In Fig. 11, editor for the learning analytics that need to be collected and is specific to the Adaptive VLE domain is presented.



```
<default> editor for concept AdaptiveSystem
  node cell layout:
    [/
       Adaptive System
       <constant>
       Autonomic Manager:
       % autonomicmanager %
       <constant>
       Managed System:
       % managedsystem %
       <constant>
    /]

  inspected cell layout:
    <choose cell model>
```

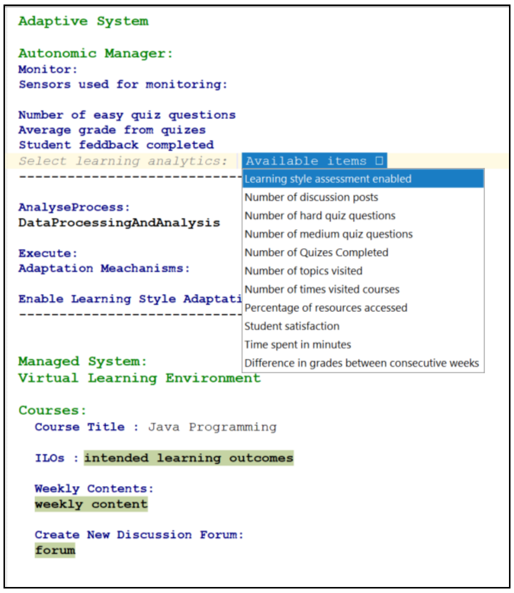
FIGURE 10. Editor for *AdaptiveSystem* concept



```
<default> editor for concept CollectLearningAnalytics
  node cell layout:
    [/
       <constant>
       (/ % items %                                          /)
          /empty cell: Press Enter to add analytics items
       ----------------------------------------------------
    /]

  inspected cell layout:
    <choose cell model>
```

FIGURE 11. Editor for concept that is specific to the AdaptiveVLE domain

The final solution-usage of the *Adaptive_VLE* language is depicted in Fig. 12. In this figure, an application of the *AdaptiveSystems* DSL editor depicted in Fig. 10 and the *Adaptive_VLE* DSL editor in Fig. 11 are presented. The "green font" main elements of adaptive systems (Fig. 10): adaptive system, autonomic manager and managed system are all present. However, for the specific learning analytics that will need to be collected, new editors (Fig. 11) were developed in the *Adaptive_VLE* DSL. These editors' User Interface (UI) contained improved drop-down menus options that appear when we hover over a particular point in the editor. The MPS feature utilised is a "transformation menu" that is used to filter-out all entries that have been already used and the "context assistant menu" of MPS. The "context assistant menu" is used to provide a menu only when the cursor is in context. Also, the *ManagedSystem* structure is depicted in Fig. 12 and the corresponding editor was developed in the derived *Adaptive_VLE* DSL.



```
Adaptive System

Autonomic Manager:
Monitor:
Sensors used for monitoring:

Number of easy quiz questions
Average grade from quizes
Student feddback completed
Select learning analytics:  Available items
----------------------------
AnalyseProcess:
DataProcessingAndAnalysis

Execute:
Adaptation Meachanisms:

Enable Learning Style Adaptati
----------------------------

Managed System:
Virtual Learning Environment

Courses:
  Course Title : Java Programming

  ILOs : intended learning outcomes

  Weekly Contents:
  weekly content

  Create New Discussion Forum:
  forum
```

Learning style assessment enabled
Number of discussion posts
Number of hard quiz questions
Number of medium quiz questions
Number of Quizes Completed
Number of topics visited
Number of times visited courses
Percentage of resources accessed
Student satisfaction
Time spent in minutes
Difference in grades between consecutive weeks

FIGURE 12. *Solution for Adaptive_VLE DSL*

Data handling languages and CAF datamapping DSL

Language structure

The *datamapping* DSL is a generic DSL for representing data structures that are stored and processed and therefore is not specific to an application domain. It is used to define features and their datatypes.

Specifically, in Fig. 13, the main structure of the language is presented in a diagrammatic form. Each of the blocks correspond to concepts in MPS JetBrains. From the figure, we can see that *Data* consist of *Features* which consist of a *FeatureMapper* concept and a *DataType* concept. The *FeatureMapper* is the linking concept that connects this language with other languages that are specific to the application domain such as the *Adaptive_VLE* in our case. The *DataType* contains all the supported types which are *CharType*, *NumberType*, *EnumerationType* and *DateType*. Both the *FeatureMapper* and the *DataType* concepts can be modified and extended in future versions of the language to include more features and more data types. This provides more dimensions of extensibility to address the needs of data science teams.

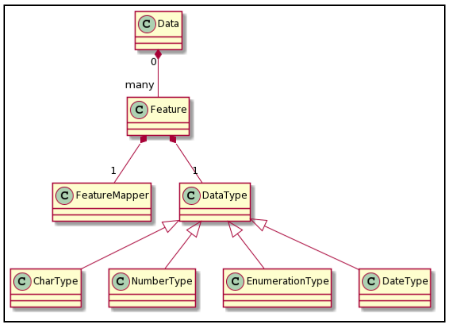
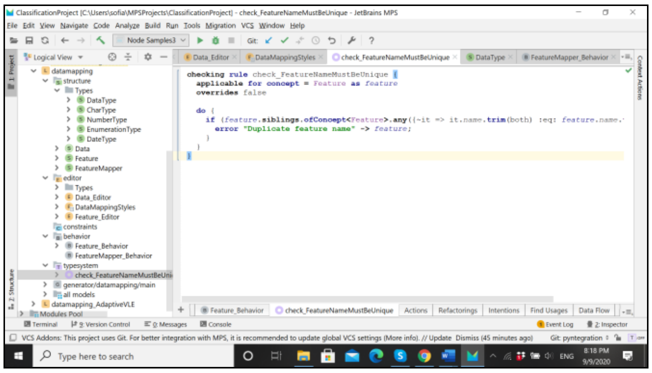


FIGURE 13. *datamapping* DSL language structure

It is interesting to note that validation has been added using checking rules for making sure that every feature defined is unique as depicted in Fig. 14. This is a very useful aspect for ensuring that the data that will be used for processing will have unique columns which is a strong requirement from BT's data science team.



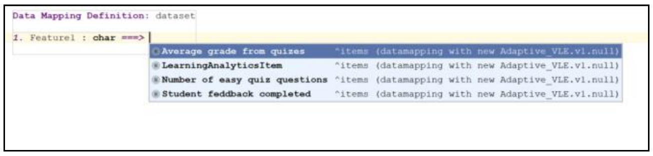
```
checking rule check_FeatureNameMustBeUnique {
  applicable for concept = Feature as feature
  overrides false

  do {
    if (feature.siblings.ofConcept<Feature>.any({~it => it.name.trim(both) :eq: feature.name.
      error "Duplicate feature name" -> feature;
    }
  }
}
```

FIGURE 14. Data validation mechanisms as part of the *datamapping* language.

More restrictions through checking rules can be added in the future to depict validation requirements from all domains involved in the system (education, data science in this particular case study).

Language usage

The *datamapping* language is an essential abstraction that is required in all applications where data is being handled. Its main benefit is the raising of abstraction level from the Excel/arff or any other required file type to a user screen such as in Fig. 15. In this figure, only names and types are defined that are required by the domain. The details of the file types are abstracted to the user. It plays vital linking role in connecting parts of adaptive systems from application-specific parts such as the *Adaptive_VLE* language to generic data processing parts such as the *CAF* language. This can be generalised for any adaptive system by replacing the application-specific part with a different language. Finally, this *datamapping* language can be reused not only by adaptive systems but by any systems that require mapping and handling of data.



```
Data Mapping Definition: dataset

1. Feature1 : char ===> |
                        Average grade from quizes     ^items (datamapping with new Adaptive_VLE.v1.null)
                        LearningAnalyticsItem         ^items (datamapping with new Adaptive_VLE.v1.null)
                        Number of easy quiz questions ^items (datamapping with new Adaptive_VLE.v1.null)
                        Student feddback completed    ^items (datamapping with new Adaptive_VLE.v1.null)
```

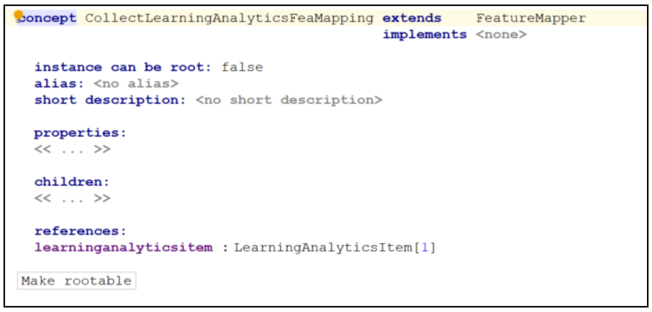
FIGURE 15. *datamapping* user interface with cross-validation of attributes

In the above figure, the features that will be used for data processing and their corresponding types are being defined. Cross-validation with the corresponding *LearningAnalyticsItems* from the *Adaptive_VLE* language is also being performed through the availability or not of in the dropdown menu. This means that the user can define only features that correspond to data that have been collected through the *Adaptive_VLE* language. More on the linking between *datamapping* and the *Adaptive_VLE* language in the following sections.

datamapping_Adaptive_VLE DSL

Language structure

The *datamapping_Adaptive_VLE* is an "adaptor" language that connects the *datamapping* DSL (language for handling data structures as described in the previous section) with the *Adaptive_VLE* DSL (application-specific language). It consists of one main concept, the *CollectLearningAnalyticsFeaMapping* that contains a reference to the *LearningAnalyticsItem* concept of the *Adaptive_VLE* DSL. The linking structures between the two languages are depicted in Fig. 16.



```
concept CollectLearningAnalyticsFeaMapping extends    FeatureMapper
                                           implements <none>

  instance can be root: false
  alias: <no alias>
  short description: <no short description>

  properties:
  << ... >>

  children:
  << ... >>

  references:
  learninganalyticsitem : LearningAnalyticsItem[1]

Make rootable
```

FIGURE 16. Concept structure of the *datamapping_Adaptive_VLE* that contains a reference to the *LearnignAnalyticsItem* concept of the *Adaptive_VLE* language.

Language usage

The *datamapping_Adaptive_VLE* language is an adaptor-linking language that is not directly accessed by the user and therefore it doesn't provide a user interface. This is implemented in MPS by not defining any *rootable* concept in the language.

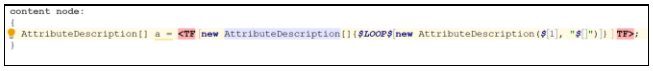
However, to demonstrate the usage of the overall language composition, we created a sandbox that includes all the data-related languages. In this sandbox, changes in the *Adaptive_VLE* defined learning analytics constructs are immediately reflected in the options available in the *datamapping* language. You can observe this if you compare the drop-down options available in Fig. 15 and the corresponding configuration of the *Adaptive_VLE* learning analytics options in Fig. 12.

All DSLs composition with CAF

Languages' composition with CAF structure

At the heart of any adaptive system, data processing through algorithms such as classification algorithms is performed. This is implemented through the *CAF* DSL language that we developed in our previous publication [10]. The *CAF* DSL takes as an input a data *arff* file (this is the type required by the *weka* libraries implementation that are being called). The *datamapping_Adaptive_VLE* language that contains the data structure adapted to particular application domain (the AdaptiveVLE in our case), is used as an input in the *CAF* DSL in order to validate that the input *arff* file is correct and according to the *datamapping* structure that was defined by the user.

This is implemented through an MPS code structure as part of MPS's automatic code generation and is depicted in the following Fig. 17.



```
content node:
{
AttributeDescription[] a = <TF [new AttributeDescription[]{$LOOP$[new AttributeDescription($[1], "$[]")]} ] TF>;
}
```

FIGURE 17. Data Validation through language composition.

In the above code, mechanisms by MPS such as the *inspector* are utilised to check the types and the names of the corresponding structures. Although there is some initial learning curve to be able to use these mechanisms, the code to implement and link elements is very concise, and it is all happening in one environment.

Languages' composition with CAF usage

To perform the composition between *CAF* and the rest of the languages, MPS's *sandbox import* mechanism was used. In *CAF* sandbox we imported the *Adaptive_VLE* sandbox as is depicted in Fig. 18. The Adaptive_VLE sandbox includes the other three languages for data collection (*Adaptive_VLE*, *datamapping*, *datamapping_Adaptive_VLE*).

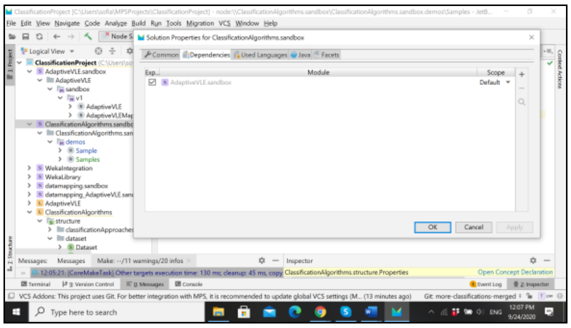
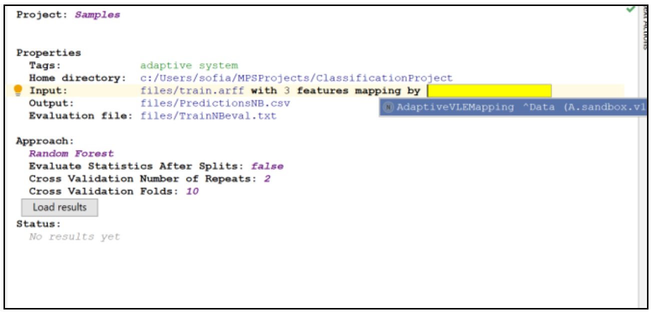

ClassificationProject [C:\Users\sofia\MPSProjects\ClassificationProject] - node:\\ClassificationAlgorithms.sandbox\ClassificationAlgorithms.sandbox.demos\Samples - JetB...
File Edit View Navigate Code Analyze Build Run Tools Migration VCS Window Help
Solution Properties for ClassificationAlgorithms.sandbox
Common
Dependencies
Used Languages
Java
Facets
Exp...
Module
Scope
AdaptiveVLE.sandbox
Default
OK
Cancel
Apply
Logical View
ClassificationProject
AdaptiveVLE.sandbox
AdaptiveVLE
sandbox
v1
AdaptiveVLE
AdaptiveVLEMap
ClassificationAlgorithms.sandbox
ClassificationAlgorithms.san
demos
Sample
Samples
WekaIntegration
WekaLibrary
datamapping.sandbox
datamapping_AdaptiveVLE.san
AdaptiveVLE
ClassificationAlgorithms
structure
classificationApproaches
dataset
Dataset
Messages: Messages
Make: --/11 warnings/20 infos
12:05:21: [CoreMakeTask] Other targets execution time: 130 ms; cleanup: 45 ms, copy
Inspector
ClassificationAlgorithms.structure.Properties
Open Concept Declaration
Terminal
9: Version Control
0: Messages
Console
Event Log
2: Inspector
VCS Addons: This project uses Git. For better integration with MPS, it is recommended to update global VCS settings (M... (13 minutes ago)
Git: more-classifications-merged
Type here to search
12:07 PM
9/24/2020

FIGURE 18. MPS sandbox for language composition diagram.

Finally, in the *CAF* sandbox, we can see the following user interface for data processing. Beyond the previous *CAF* interface, a new field has been added that validates the *datamapping* language instance (*AdaptiveVLEMapping*) in comparison with input data file (*arff format*) that is used and is declared by its location (*files/train.arff*).



```
Project: Samples


Properties
  Tags:               adaptive system
  Home directory:     c:/Users/sofia/MPSProjects/ClassificationProject
  Input:              files/train.arff with 3 features mapping by
  Output:             files/PredictionsNB.csv                 AdaptiveVLEMapping ^Data (A.sandbox.vl
  Evaluation file:    files/TrainNBeval.txt

Approach:
  Random Forest
  Evaluate Statistics After Splits: false
  Cross Validation Number of Repeats: 2
  Cross Validation Folds: 10
 Load results
Status:
  No results yet
```

FIGURE 19. **Usage of composition of all languages.**

All the above sections described in detail the composition of four languages that were developed to provide a systematic design and development for an adaptive VLE. The above modularity enables generalisation to other adaptive systems application domains by modifying only a small part and is being described in the following section.

IV. BENEFITS OF THE DEVELOPED *AdaptiveSystems* DSL – GUIDANCE FOR FUTURE WORK

The main benefits from the developed *AdaptiveSystems* DSL is its applicability to several application domains through its higher-abstraction level, language composition and reusability and its strong validation characteristic.

A. APPLICABILITY TO OTHER APPLICATION DOMAINS

Summarizing, in this paper we proposed a high-level description language for adaptive systems based on the MAPE-K loop, the *AdaptiveSystems* DSL. We also provided a method for instantiating this language in several application domain taking as an example the AdaptiveVLE application domain.

A set of DSL languages have been used and MPS JetBrains's strong language composability feature has been exploited.

These languages were as follows:

- *Adaptive_VLE:* application domain language for configuring learning analytics that will be collected.

- *datamapping:* language for high-level description of the data.

- *datamapping_Adaptive_VLE:* adaptor language that links *datamapping* to a specific application domain.

- *CAF (Classification Algorithms Framework):* language for data processing with validation that the correct features are being processed.

To reuse this in a different application domain only two languages from the above need to be rewritten/modified and they are the ones that are related to the application domain: *Adaptive_VLE* and *datamapping_Adaptive_VLE.*

Generalising those, we will name them *DataCollection* and *datamapping_DataCollection* languages and the methodology to construct them is as follows:

- *DataCollection* language: The *Adaptive_VLE* consists of a domain-specific user interface for defining the data that need to be collected and it produces XML file output. This XML is used as an input to configure the VLE website development. It is straightforward to apply the same method when the application domain consists of the development of a website that is used for data collection. However, the XML format output can used by many other environments as an input and therefore can ensure interoperation with other tools beyond web-development environments.

- *datamapping_DataCollection* language: The *datamapping_Adaptive_VLE* connects the *LearningAnalyticsItem* concept from the configuration of the data that will be collected. That can be declared as a child concept of the main concepts of the *DataCollection* language and be referenced by the *datamapping_DataCollection*. All the other structures that are used by the other languages and for their composition remain the same.

.

It is important to emphasize that this use of a domain-specific language environment seems to attempt to generalise and therefore contradict to domain-specific aspect of these environments. However, this is only appearing to be the case. It has been attempted before as one of the ways to use the MPS domain-specific language environment for framework development due to the strong code generation features and language composition structures [12].

B. OTHER BENFICIAL CHARACTERISTICS

The set of languages that were used and their composition enables strong validation for the data and their types that are being collected and processed.

It also provides a higher-level of abstraction for the data which is closer to the domain user. Consequently, the level of complexity that a user must deal with is less, as the abstraction level is higher, enabling a system overview of adaptive systems design and development. This was all "enforced" using the higher-abstraction level language for adaptive systems, the *AdaptiveSystems* DSL.

V. EVALUATION

We evaluated the *AdaptiveSystems* DSL according to the Quality characteristics defined in the paper [38] and using the feedback from MPS's language developers and BT's data scientists in the first instance.

The decision to use this approach was after extensive literature review on the evaluation methods for domain-specific languages. We mainly used the information that was provided in the survey [39] and specifically for the evaluation part in Table 4. Four main methods were identified for evaluating domain-specific languages: Case studies as in [40] which didn't include questionnaires/feedback and the results were presented as logical technical arguments; Questionnaire & Experiments as in [41] which were tailored to the particular application domain and didn't contain domain-specific language metrics; Usability Testing as in [42] which contained general usability methods again not tailored to language development; Use Case as in which again focuses on application-specific characteristics. Overall and to the best of our knowledge, the approach we followed was the best for evaluating domain-specific language tailored to the language's characteristics.

A. EVALUATION PROCESS

We evaluated the developed DSL from the domain perspective of the adaptive systems developer. The reason for not including educators is that we have evaluated that part of the languages in our previous publications and the emphasis and innovation here lies in the general adaptive systems development or equally stated providing AI in context. In this current paper, the main aim is to provide tools/automation for the adaptive systems developer and the language developer. For the evaluation we followed the quality characteristics as defined in [43].

Specifically, a questionnaire was constructed and distributed online. The questionnaire collected quantitative information by asking participants to rate the language using a score from 0 to 10 (0 stands for not addressed at all and 10 stands for addressing the issue perfectly). The questionnaire also contained open-ended questions so that qualitative information could be collected. An example of an open-ended question for the data science perspective was as follows: *"(Functional Suitability) Does the language contain all the data validation functionality you would like to have. If you answered no, what should be included?"*. We also have some mixed scoring and open-ended questions such as the following in all perspectives *"(Usability) How much would you rate (in a scale of 0 to 10) the language as regards to its usability? If you rated the usability over 5, can you please give a couple of good points that you found? If you rated the usability under 5, can you please add suggestions for improvements?"*.

The sample of the responses collected consisted of a total of 25 participants of which 15 were BT's data scientists, and 10 were DSL language developers.

B. RESULTS - DISCUSSION

The results from the two different users were very similar between them and therefore are collectively presented as follows:

Functional suitability: The language scored very high according to its suitability for all the functionality required by all and especially the data scientists. The data validation was an impressive characteristic and more validation options were suggested such as restricting users with only permitted options in all parts of the interface. It also scored high by the language developers as *"it provided the necessary structure"* that enables the development of DSLs for adaptive systems.

Usability: Regarding usability, the language scored medium as it takes some time to get used to the different features.

Reliability: The language was more reliable than the typical adaptive system operation. However, many suggestions to add validation characteristics were made by BT's engineers.

Productivity: The use of this language enabled a quick turn-around time for an adaptive system that included the use of two screens and one integrated development environment.

Compatibility: The Java code generation enabled extensive deployment options.

Expressiveness: The language is very expressive. However, more work could be done to improve the interface and therefore the user experience.

Reusability: This set of languages provides an excellent reusability element as it can be used as part of any adaptive system.

Maintainability: It scored medium for maintainability and a separate interface for extensibility was the dominating suggestion.

It is anticipated that this method will be applicable in many research and commercial projects at BT and other domains such as health care.

The scoring for all individual elements is included in the Excel file with the results in Fig. 20.

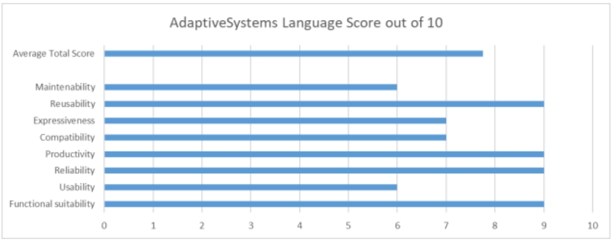


FIGURE 20. Evaluation through feedback by BT's engineers.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a generic approach to the design and development of an adaptive system. We developed a framework for adaptive systems based on the main elements of the MAPE-K loop. From the generic *AdaptiveSystems* DSL, we derived (MPS's language extension feature) the

Adaptive_VLE DSL for the adaptive VLE case study. We also developed several reusable languages for the various stages required by adaptive systems mainly tailored to data handling such as the *datamapping*, *datamapping_Adaptive_VLE and CAF*. Following the proposed approach, the enforcement of the MAPE-K blueprint architecture, a full path to implementation and data validation were ensured. We evaluated the resulting composed language through feedback from BT's adaptive systems and MPS's language engineers. Finally, we believe that this work significantly contributes to the effective design and development of adaptive systems which is a significant part of the greater problem of providing AI in context.

In the immediate future, we plan to extend the developed language to more application domains such as health care, etc. and to IoT applications.

Within our long-term plans, many extensions can follow this initial work ranging from adding adaptivity patterns, extending the architecture to decentralised approaches to adding formal methods capabilities to ensure properties such as safety, fairness, robustness.

ACKNOWLEDGMENT

We would like to acknowledge BT for funding this research through financial support to the University and in-kind support of staff time and effort.

REFERENCES

[1] B. H. C. Cheng *et al.*, "Software engineering for self-adaptive systems: A research roadmap," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009, vol. 5525 LNCS, pp. 1–26, doi: 10.1007/978-3-642-02161-9_1.

[2] P. Lalanda, J. A. McCann, and A. Diaconescu, *Autonomic Computing*. London: Springer London, 2013.

[3] "Making Sense of Change Management: A Complete Guide to the Models, Tools and ... - Esther Cameron, Mike Green - Βιβλία Google." https://books.google.co.uk/books?hl=el&lr=&id=LX-5DwAAQBAJ&oi=fnd&pg=PP1&dq=change+management+models&ots=v-fHPi1Pzr&sig=SigKvbpUhbXX_gZPQO81a01rxpo&redir_esc=y#v=onepage&q=change management models&f=false (accessed Oct. 20, 2020).

[4] S.-M. Hossein and G. A., "A language for high-level description of adaptive web systems," *J. Syst. Softw.*, vol. 81, no. 7, pp. 1196–1217, Jul. 2008, doi: 10.1016/J.JSS.2007.08.033.

[5] C. R. Teeneti *et al.*, "System-Level Approach to Designing a Smart Wireless Charging System for Power Wheelchairs," *IEEE Trans. Ind. Appl.*, pp. 1–1, 2021, doi: 10.1109/TIA.2021.3093843.

[6] A. Moin, A. Badii, and S. Günnemann, "A Model-Driven Engineering Approach to Machine Learning and Software Modeling," Jul. 2021, Accessed: Jul. 18, 2021. [Online]. Available: https://arxiv.org/abs/2107.02689v1.

[7] S. Al-Fedaghi and M. Alsaraf, "High-Level Description of Robot Architecture," *IJACSA) Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 10, 2020, Accessed: Jul. 18, 2021. [Online]. Available: www.ijacsa.thesai.org.

[8] L. Shen, X. Chen, R. Liu, H. Wang, and G. Ji, "Domain-Specific Language Techniques for Visual Computing: A Comprehensive Study," *Arch. Comput. Methods Eng. 2020 284*, vol. 28, no. 4, pp. 3113–3134, Oct. 2020, doi: 10.1007/S11831-020-09492-4.

[9] I. Portugal, P. Alencar, and D. Cowan, "A Survey on Domain-Specific Languages for Machine Learning in Big Data," Feb. 2016, Accessed: Jul. 18, 2021. [Online]. Available: https://arxiv.org/abs/1602.07637v2.

[10] S. Meacham, V. Pech, and D. Nauck, "Classification Algorithms Framework (CAF) to Enable Intelligent Systems Using JetBrains MPS Domain-Specific Languages Environment," *IEEE Access*, vol. 8, pp. 14832–14840, 2020, doi: 10.1109/ACCESS.2020.2966630.

[11] "MPS: The Domain-Specific Language Creator by JetBrains." https://www.jetbrains.com/mps/ (accessed Nov. 01, 2019).

[12] M. Voelter and V. Pech, "Language modularity with the MPS language workbench," in *Proceedings - International Conference on Software Engineering*, 2012, pp. 1449–1450, doi: 10.1109/ICSE.2012.6227070.

[13] S. Meacham, V. Pech, and D. Nauck, "Classification Algorithms Framework (CAF) to Enable Intelligent Systems Using JetBrains MPS Domain-Specific Languages Environment," *IEEE Access*, vol. 8, pp. 14832–14840, 2020, doi: 10.1109/ACCESS.2020.2966630.

[14] S. Meacham, "Towards Self-Adaptive IoT Applications: Requirements and Adaptivity Patterns for a Fall-Detection Ambient Assisting Living Application," in *Components and Services for IoT Platforms*, Cham: Springer International Publishing, 2017, pp. 89–102.

[15] P. Lalanda, J. A. McCann, and A. Diaconescu, "Future of Autonomic Computing and Conclusions," 2013, pp. 263–278.

[16] M. Mikalsen, J. Floch, E. Stav, N. Paspallis, G. A. Papadopoulos, and P. A. Ruiz, "Putting context in context: The role and design of context management in a mobility and adaptation enabling middleware," in *Proceedings - IEEE International Conference on Mobile Data Management*, 2006, vol. 2006, doi: 10.1109/MDM.2006.129.

[17] S. M. Sadjadi and P. K. McKinley, "ACT: An adaptive CORBA template to support unanticipated adaptation," in *Proceedings - International Conference on Distributed Computing Systems*, 2004, vol. 24, pp. 74–83, doi: 10.1109/icdcs.2004.1281570.

[18] S. W. Cheng and D. Garlan, "Stitch: A language for architecture-based self-adaptation," *J. Syst. Softw.*, vol. 85, no. 12, pp. 2860–2875, Dec. 2012, doi: 10.1016/j.jss.2012.02.060.

[19] D. B. Abeywickrama, N. Hoch, and F. Zambonelli, "SimSOTA: Engineering and simulating feedback loops for self-adaptive systems," in *ACM International Conference Proceeding Series*, 2013, pp. 67–76, doi: 10.1145/2494444.2494446.

[20] F. Křikava, P. Collet, and R. B. France, "ACTRESS: Domain-specific modeling of self-adaptive software architectures," in *Proceedings of the ACM Symposium on Applied Computing*, 2014, pp. 391–398, doi: 10.1145/2554850.2555020.

[21] M. D'Angelo, M. Caporuscio, and A. Napolitano, "Model-driven engineering of decentralized control in cyber-physical systems," in *Proceedings - 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems, FAS*W 2017*, Oct. 2017, pp. 7–12, doi: 10.1109/FAS-W.2017.113.

[22] F. Fleurey and A. Solberg, "A domain specific modeling language supporting specification, simulation and execution of dynamic adaptive systems," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009, vol. 5795 LNCS, pp. 606–621, doi: 10.1007/978-3-642-04425-0_47.

[23] A. Bucchiarone, A. Cicchetti, and M. De Sanctis, "Towards a domain specific language for engineering collective adaptive systems," in *Proceedings - 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems, FAS*W 2017*, Oct. 2017, pp. 19–26, doi: 10.1109/FAS-W.2017.115.

[24] F. Alvares, E. Rutten, and L. Seinturier, "A domain-specific language for the control of self-adaptive component-based architecture," *J. Syst. Softw.*, vol. 130, pp. 94–112, Aug. 2017, doi: 10.1016/j.jss.2017.01.030.

[25] P. Arcaini, R. Mirandola, E. Riccobene, and P. Scandurra, "MSL: A pattern language for engineering self-adaptive systems," *J. Syst. Softw.*, vol. 164, p. 110558, Jun. 2020, doi:

10.1016/j.jss.2020.110558.

[26] D. Weyns *et al.*, "On patterns for decentralized control in self-adaptive systems," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013, vol. 7475 LNCS, pp. 76–107, doi: 10.1007/978-3-642-35813-5_4.

[27] J. Bocanegra, J. Pavlich-Mariscal, and A. Carrillo-Ramos, "DMLAS: A Domain-Specific Language for designing adaptive systems," in *2015 10th Colombian Computing Conference, 10CCC 2015*, Nov. 2015, pp. 47–54, doi: 10.1109/ColumbianCC.2015.7333411.

[28] S. Meliá, C. Cachero, J. M. Hermida, and E. Aparicio, "Comparison of a textual versus a graphical notation for the maintainability of MDE domain models: an empirical pilot study," *Softw. Qual. J.*, vol. 24, no. 3, pp. 709–735, Sep. 2016, doi: 10.1007/s11219-015-9299-x.

[29] M. Berndtsson and T. Svahn, "Strategies for Scaling Analytics: A Nontechnical Perspective."

[30] M. Berndtsson, C. Lennerholt, T. Svahn, and P. Larsson, "13 organizations' attempts to become data-driven," *Int. J. Bus. Intell. Res.*, vol. 11, no. 1, pp. 1–21, Jan. 2020, doi: 10.4018/IJBIR.2020010101.

[31] K. Sandkuhl, "Putting AI into context-Method support for the introduction of artificial intelligence into organizations," in *Proceedings - 21st IEEE Conference on Business Informatics, CBI 2019*, Jul. 2019, vol. 1, pp. 157–164, doi: 10.1109/CBI.2019.00025.

[32] "What is ArchiMate?" https://www.visual-paradigm.com/guide/archimate/what-is-archimate/ (accessed Oct. 20, 2020).

[33] V. Beaudouin *et al.*, "Flexible and Context-Specific AI Explainability: A Multidisciplinary Approach," *SSRN Electron. J.*, Apr. 2020, doi: 10.2139/ssrn.3559477.

[34] Z. Chaczko, M. Kulbacki, G. Gudzbeler, M. Alsawwaf, I. Thai-Chyzhykau, and P. Wajs-Chaczko, "Exploration of Explainable AI in Context of Human-Machine Interface for the Assistive Driving System," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Mar. 2020, vol. 12034 LNAI, pp. 507–516, doi: 10.1007/978-3-030-42058-1_42.

[35] "An architectural blueprint for autonomic computing," 2005.

[36] M. Luckey, B. Nagel, C. Gerth, and G. Engels, "Adapt cases: Extending use cases for adaptive systems," in *Proceedings - International Conference on Software Engineering*, 2011, pp. 30–39, doi: 10.1145/1988008.1988014.

[37] S. Meacham, V. Pech, and D. Nauck, "AdaptiveVLE: an integrated framework for personalised online education using MPS JetBrains domain-specific modelling environment," *IEEE Access*, pp. 1–1, 2020, doi: 10.1109/ACCESS.2020.3029888.

[38] M. Challenger, G. Kardas, and B. Tekinerdogan, "A systematic approach to evaluating domain-specific modeling language environments for multi-agent systems," *Softw. Qual. J.*, vol. 24, no. 3, pp. 755–795, Sep. 2016, doi: 10.1007/s11219-015-9291-5.

[39] A. J. Salman, M. Al-Jawad, and W. Al Tameemi, "Domain-Specific Languages for IoT: Challenges and Opportunities," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1067, no. 1, p. 012133, Feb. 2021, doi: 10.1088/1757-899X/1067/1/012133.

[40] J. Verriet *et al.*, "Virtual Prototyping of Large-scale IoT Control Systems using Domain-specific Languages," pp. 229–239, May 2019, doi: 10.5220/0007250402290239.

[41] "A User-Oriented Language for Specifying Interconnections Between Heterogeneous Objects in the Internet of Things | IEEE Journals & Magazine | IEEE Xplore." https://ieeexplore.ieee.org/document/8606175 (accessed Aug. 17, 2021).

[42] T. Eterovic, E. Kaljic, D. Donko, A. Salihbegovic, and S. Ribic, "An Internet of Things visual domain specific modeling language based on UML," *2015 25th Int. Conf. Information, Commun. Autom. Technol. ICAT 2015 - Proc.*, Nov. 2015, doi: 10.1109/ICAT.2015.7340537.

[43] M. Challenger, G. Kardas, and B. Tekinerdogan, "A systematic approach to evaluating domain-specific modeling language environments for multi-agent systems," *Softw. Qual. J.*, vol. 24, no. 3, pp. 755–795, Sep. 2016, doi: 10.1007/s11219-015-9291-5.

SOFIA MEACHAM *Dr Sofia Meacham received her* Diploma in Computer and Informatics Engineering, in 1994, and her PhD degree, in 2000, from the University of Patras, Greece. She is currently a Senior Lecturer in Software Engineering at Bournemouth University, U.K. Her PhD research interests fell in the area of system-level design for embedded systems, and include specification techniques for complex embedded telecommunication systems, hardware-software co-design, formal refinement techniques, and reuse practices. Dr. Meacham has been working in EU-funded projects as a researcher/embedded software engineer both in Industry and in University since 1995, and have accomplished a large amount of teaching experience in several institutions (UK and Greece) since 2000. Her latest research interests involve methodologies (processes, tools and methods) to improve the design and development of systems such as model-based design, domain-specific modelling for several applications from business processes to education. She has strong links with British Telecom Research Headquarters in Adastral Park and currently working in cutting-edge research in Explainable AI.

VACLAV PECH Vaclav Pech is a seasoned software developer and a programming enthusiast with 22 years of Java development and consultancy experience. He received his Master's Degree in Computer Science in 1999 from the Faculty of Mathematics and Physics of the Charles University in Prague. Since then he has participated as a developer and consultant in various projects across Europe working mainly with server-side Java technologies and domain-specific languages. Currently he is involved in the MPS project with JetBrains.

DETLEF NAUCK Detlef Nauck is Chief Research Scientist for Data Science with BT's Research and Innovation Division located at Adastral Park, Ipswich, UK. He is leading a group of international scientists working on research into Data Science, Machine Learning and AI. Detlef focuses on establishing best practices in Data Science for conducting analytics professionally and responsibly leading to new ways of analyzing data for achieving better insights. Part of his role is leading the initiative on the development and use of responsible and ethical AI in the company. Detlef is a computer scientist by training and holds a PhD and a Postdoctoral Degree (Habilitation) in Machine Learning and Data Analytics. He is a Visiting Professor at Bournemouth University and a Private Docent at the Otto-von-Guericke University of Magdeburg, Germany. He has published 3 books, over 120 papers, holds 10 patents and has 30 active patent applications.