Research on Reference Architectures for Self-Adaptive Systems

A Systematic Mapping Protocol

Author:
Lina Garcés, University of São Paulo (USP)

January 2020

Presentation of the Document

This document contains the protocol of the systematic mapping study conducted to investigate reference architectures for self-adaptive systems.

This document was prepared for supporting information contained in the manuscript titled "*Architectural Solutions for Self-Adaptive Systems"* authored by *Lina Garcés (Ph.D.), Silverio Martínez-Fernández (Ph.D.), Valdemar V.G. Neto (Prof. Dr.), and Elisa Yumi Nakagawa (Prof. Dr.).*

Contents

1.Introduction

A reference architecture is “an architecture that encompasses the knowledge about how to design concrete architectures of systems of a given application [or technological] domain; therefore, it must address the business rules, architectural styles (sometimes also defined as architectural patterns that address quality attributes in the reference architecture), best practices of software development (for instance, architectural decisions, domain constraints, legislation, and standards), and the software elements that support development of systems for that domain. All of this must be supported by a unified, unambiguous, and widely understood domain terminology” [Nakagawa et al. 2011].

The motives behind the adoption of reference architectures are: to facilitate reuse, and thereby harvest potential savings through reduced cycle times, cost, risk and increased quality [Cloutier et al. 2010]; to help with the evolution of a set of systems that stem from the same RA [Galster and Avgeriou 2011]; and to ensure standardization and interoperability [Angelov et al. 2012]. Due to this, RAs are becoming a key asset of organizations [Cloutier et al. 2010].

From this perspective, RAs have become an important approach in software development projects in the industry. Specifically, in times when the size and complexity of software systems, together with critical time-to-market needs, demand new software engineering approaches to software development.

Self-adaptive Systems (SaS) are complex systems that have the ability to modify their behavior or configuration at runtime, reacting to changes in their environment, new goals or due to failures [Oreizy et al. 1999]. Such modifications should occur without affecting the accomplishment of the SaS missions neither their reliability, security, safety, nor other quality attributes. Hence, engineering SaS is a time- and effort-consuming task, bringing important challenges mainly to design their architecture; however, there is a lack of solutions that guide such design.

We decided to conduct a systematic mapping study (SMS) to have a broad overview of existing RAs for SaS. For this, we followed the guidelines in [Petersen et al. 2015]. With this SMS, we intend to mining from RAs the most recurrent architectural solutions for SaS. This document describes the protocol of the SMS that was conducted for identifying the RAs for SaS and perform architectural solutions extraction. Moreover, the results of conducting the mapping are also presented.

2.Planning the Mapping

Following, we define the SMS protocol. It consists of the five steps aforementioned: definitions of research questions, the strategy that will be used to search for primary studies, the study selection criteria and procedures, keywording using abstracts, and the data extraction and mapping process.

Goals and Research Questions

Goal: To identify recurrent architectural solutions for SaS that are contained in RAs.

Research Questions:

1. *What reference architectures for SaS have been reported in the scientific literature?*
2. *What are the main software building blocks that compose SaS architectures?*
3. *How SaS architectures vary depending on the adaptivity capabilities and control characteristics of these systems?*

Search Strategy

Scientific databases: We used Scopus, Web of Science, IEEE Xplore, ACM Digital Library, ScienceDirect, and SpringerLink scientific data libraries as recommended by [Dieste and Padua 2007] and [Dyba et al. 2007] for their efficiency at searching research work in the software engineering area.

Search string: We defined the following string to be executed in the scientific data libraries:

("reference architecture?")
AND
("software architecture?" or "software structure?" or "software design?" or "system architecture?" or "system structure?" or "system design?")

We opted to do not limit the search only to SaS, since the term "self-adaptive systems" could not be explicitly used in the RAs. Therefore, we decided to look for adaptivity properties in existing RAs.

Studies Selection: The following criteria were used to decide the inclusion or exclusion of primary studies in our mapping study.

Inclusion Criteria (IC):
IC1 - The work is mainly focused on a reference architecture for software systems of any domain.
IC2 - Software systems considered in the reference architecture have adaptivity properties (e.g., self-adaptivity, self-configuration, self-organization, etc.)

Exclusion Criteria (EC):

EC1 - The topic of the study is NOT focused on reference architectures for software systems but in other area and the term "reference architecture" just appear incidentally. Although the work may be about a reference architecture or an approach to support a reference architecture related practice, it is NOT mainly used for software systems.
EC2 - The reference architecture presented in the study does not deal with software systems that have adaptivity characteristics.
EC3 - The study was not peer-reviewed.
EC4 - The study is not accessible even after contacting the authors.
EC5 - The study is not completely written in English.

3.Search Conduction

This SMS was conducted from January to July 2018 by researchers from both industry and academia and with experience in self-adaptive systems, reference architectures and software architectures, besides their experience in researching, and conducting and updating a number of SMS and systematic literature review (SLR). To support this selection process, we used JabRef (http://www.jabref.org), a reference management tool.

Figure 1 depicts the steps of the selection process. By adapting the search string for each database and considering the search on title, abstract, and keywords, we obtained a total of **989** studies and removing the duplicated studies, **589** studies remained. After the first selection where we applied the selection criteria **IC1** on title, abstract, and keywords, **183** studies were selected. After reading the full text of these studies and applying the selection criteria **IC1** again, a final set of **142** primary studies were selected. Besides that, a snowballing inspection on the list of references of each selected study made us possible to include other **19** relevant studies, totaling 161 studies from which it was possible to identify **161** unique reference architectures.

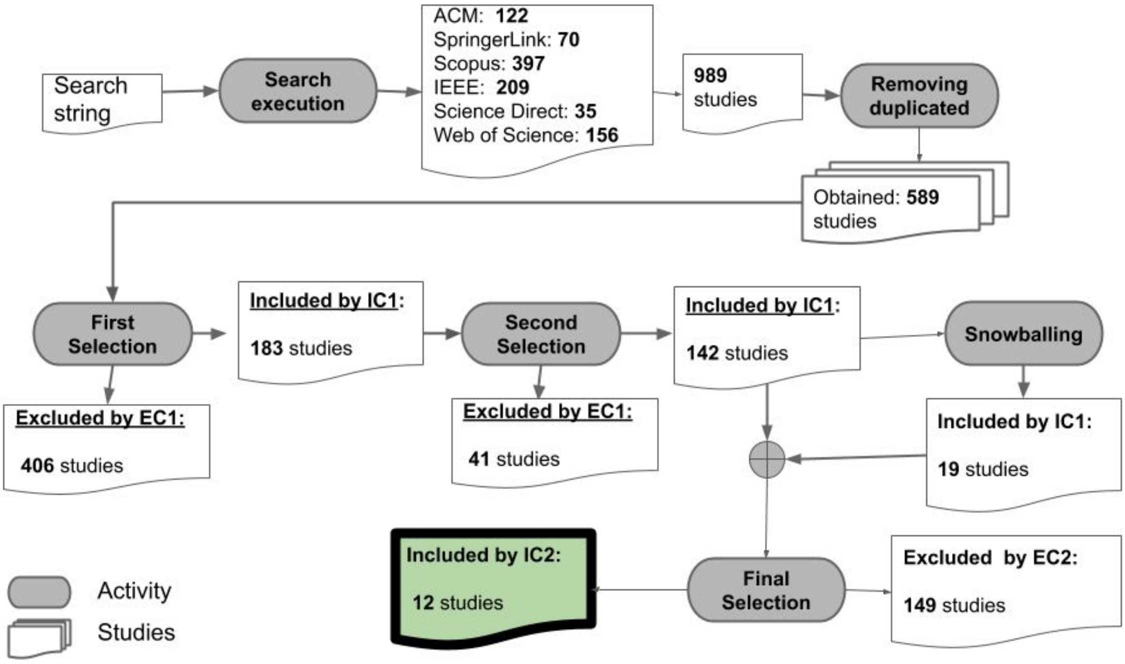


Figure 1. The selection process of studies reporting reference architectures for SaS following [Petersen et al. 2015].

After these steps, we analyzed the 161 reference architectures and applied the selection criteria **IC2,** choosing only those RAs oriented to software systems with adaptivity characteristics. Finally, we identified **12 RAs for SaS**, listed as follows.

Selected Studies

RA1. Barbara Hayes-Roth and Jan Eric Larsson. A domain-specific software architecture for a class of intelligent patient monitoring agents, Journal of Experimental and Theoretical Artificial Intelligence, vol. 8, pp. 149-171 (1996).

RA2. IBM Corporation. An architectural blueprint for autonomic computing. Technical report, IBM Corporation (2006).

RA3. Berhane Zewdie, and C. R. Carlson. Adaptive component paradigm for highly configurable business components. In EIT'06, pp. 185-190 (2006).

RA4. Lei Liu, Stefan Thanheiser, Harmut Schmeck. A Reference Architecture for Self-organizing Service-Oriented Computing. In U. Brinkschulte, T. Ungerer, C. Hochberger and R. G. Spallek (eds.) Architecture of Computing Systems – ARCS, pp. 205-219, Springer Berlin Heidelberg (2008).

RA5. Lorena C. Bueno and Gabriel Tamura. A Reference Architecture for Component-Based Self-Adaptive Software Systems. Master project. Department of

Information and Communication Technologies Faculty of Engineering. ICESI University, pp.70 (2012).

RA6. Frank J. Affonso and Elisa Yumi Nakagawa. A Reference Architecture Based on Reflection for Self-Adaptive Software, In SBCARS'13, pp. 129–138 (2013).

RA7. Jennifer Perez, Jessica Diaz, Carlos Vidal, Daniel Rodriguez, Diego Fernandez. Self-Balancing Distributed Energy in Power Grids: An Architecture Based on Autonomic Computing. In HICSS'14, pp. 2398–2407 (2014).

RA8. Ricardo Sanz, Carlos Hernandez, Julita Bermejo, Manuel Rodriguez, Ignacio Lopez. Improved Resilience Controllers Using Cognitive Patterns. In IFAC'14, pp. 683-688 (2014).

RA9. Hossein Tajalli, Nenad Medvidovic. A Reference Architecture for Integrated Development and Run-Time Environments. Software - Practice and Experience, vol. 44, pp. 299-316 (2012).

RA10. Uwe ABmann, Sebastian Gotz, Jean-Marc Jezequel, Brice Morin, Mario Trapp. A Reference Architecture and Roadmap for Models@run.time Systems. In: Bencomo, N., France, R., Cheng, B., ABmann, U. models@run.time. Foundations, Applications, and Roadmaps. Springer International Publishing. pp. 1-18 (2014).

RA11. Victor Braberman, Nicolas D´ıpolito, Jeff Kramer, Daniel Sykes, Sebastian Uchitel. MORPH: A Reference Architecture for Configuration and Behaviour Self-adaptation. In CTSE'15, pp. 9-16 (2015).

RA12. Jesus M.T. Portocarrero, Flavia C. Delicato, Paulo F. Pires, Bruno Costa, Wei Li, Weisheng Si, Albert Y. Zomaya. RAMSES: A new reference architecture for self-adaptive middleware in Wireless Sensor Networks, Ad Hoc Networks, Vol. 55, pp. 3-27 (2017).

4.Data Extraction

To obtain evidence to answer our RQs we read the full paper and extracted the following information of each paper:

- Author(s), Author(s) institution, year of publication, the title of the study
- Data library (e.g., Scopus, SpringerLink) and venue (e.g., conference, journal) where the study was published
- The study's keywords
- The application domain of the reference architecture
- Adaptivity properties and quality attributes requirements of software systems considered by the reference architecture
- Architectural patterns used by the reference architecture
- Whether the RA is oriented to industry

- Software structures, managed system, managing system, and control loops proposed in the RA
- SaS missions/goals described in the RA
- General description of the RA and the proposed solutions
- The rationale behind the proposed solution in the RA
- RA liabilities

To register the extracted information we used a LibreOffice spreadsheet. Appendix A contains the information that was extracted for each study.

5.Data Analysis

For data analysis, we used qualitative and narrative synthesis methods as recommended in [Felizardo et al. 2017].

References

[Angelov et al. 2013] Angelov, S., Trienekens, J. and Kusters, R., 2013. Software reference architectures-exploring their usage and design in practice. In Software Architecture. pp.17–24.

[Cloutier et al. 2010] Cloutier, R., Muller, G., Verma, D., Nilchiani, R., Hole, E. and Bone, M., 2010. The Concept of Reference Architectures. Systems Engineering, 13(1), pp.14–27.

[Dieste and Padua 2007] Dieste, O. and Padua, A.G., 2007. Developing Search Strategies for Detecting Relevant Experiments for Systematic Reviews. In First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007). IEEE, pp. 215–224.

[Dyba et al. 2007] Dyba, T., Dingsoyr, T. and Hanssen, G.K., 2007. Applying Systematic Reviews to Diverse Study Types: An Experience Report. In First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007). IEEE, pp. 225–234.

[Felizardo et al. 2017] Felizardo, K.T ; Nakagawa, E..Y., Fabbri, S.C.P.F., Ferrari, F., 2017. Revisão sistemática da literatura em Engenharia de Software: teoria e prática. 1 ed., Elsevier.

[Galster and Avgeriou 2011] Galster, M. and Avgeriou, P., 2011. Empirically-grounded reference architectures: a proposal. In Proceedings of the joint ACM SIGSOFT conference -- QoSA and ACM SIGSOFT symposium -- ISARCS on Quality of software architectures -- QoSA and architecting critical systems -- ISARCS. pp. 153–157.

[Nakagawa et al. 2011] Nakagawa, E.Y. 2011. Ramodel: A reference model for reference architectures. In Joint Working IEEE/IFIP Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA). pp. 297–301.

[Oreizy et al. 1999] Peyman Oreizy, Michael M. Gorlick, Richard N. Taylor, Dennis Heimbigner, Gregory Johnson, Nenad Medvidovic, Alex Quilici, David S. Rosenblum, and Alexander L. Wolf. An Architecture-Based Approach to Self-Adaptive Software. IEEE Intelligent Systems. vol.14, 3, pp. 54-62. (1999).

[Petersen et al. 2015] Petersen, K., Vakkalanka, S., Kuzniarz, L., 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. Information and Software Technology 64, 1 – 18.

Appendix

Extracted Data from Reference Architectures of Self-Adaptive Systems

---

DATA EXTRACTION

Institution	Computer science department, Stanford University, USA
Author	Barbara Hayes-Roth and Jan Eric Larsson
Year	1996
Title	A domain-specific software architecture for a class of intelligent patient monitoring agents

	ACM	IEEE	SCOPUS	WEB OF SCIENCE	COMPENDEX	SCIENCE DIRECT	SPRINGER	MANUAL
Source			x	x				

Venue	Journal of Experimental and Theoretical Artificial Intelligence

	Conference	Journal	Book chapter	Report
Type of paper		x		

Keywords	Domain-specific software architecture	adaptive intelligent systems	monitoring	fault diagnosis	medical diagnosis	

Application domain	Health care software systems – intelligent patient monitoring system

	TRL1	TRL2	TRL3	TRL4	TRL5	TRL6	TRL7	TRL8	TRL9
TRL level				x					

	General level		Major level				Primitive level		Others
Adaptive properties	Self-adaptive	Self-managed	Self-configuring	Self-healing	Self-protecting	Self-optimizing	Self-awareness	Context-awareness	
							x	x	

Quality attributes	performance eff	dynamic adaptat	robustness	flexibility	interoperability				

Architectural patterns	Layers	pipe and filters	blackboard						

Industry-oriented (name of industry)	funded by NASA

Structure adapted	control model in order to select best cognitive and physical behaviors

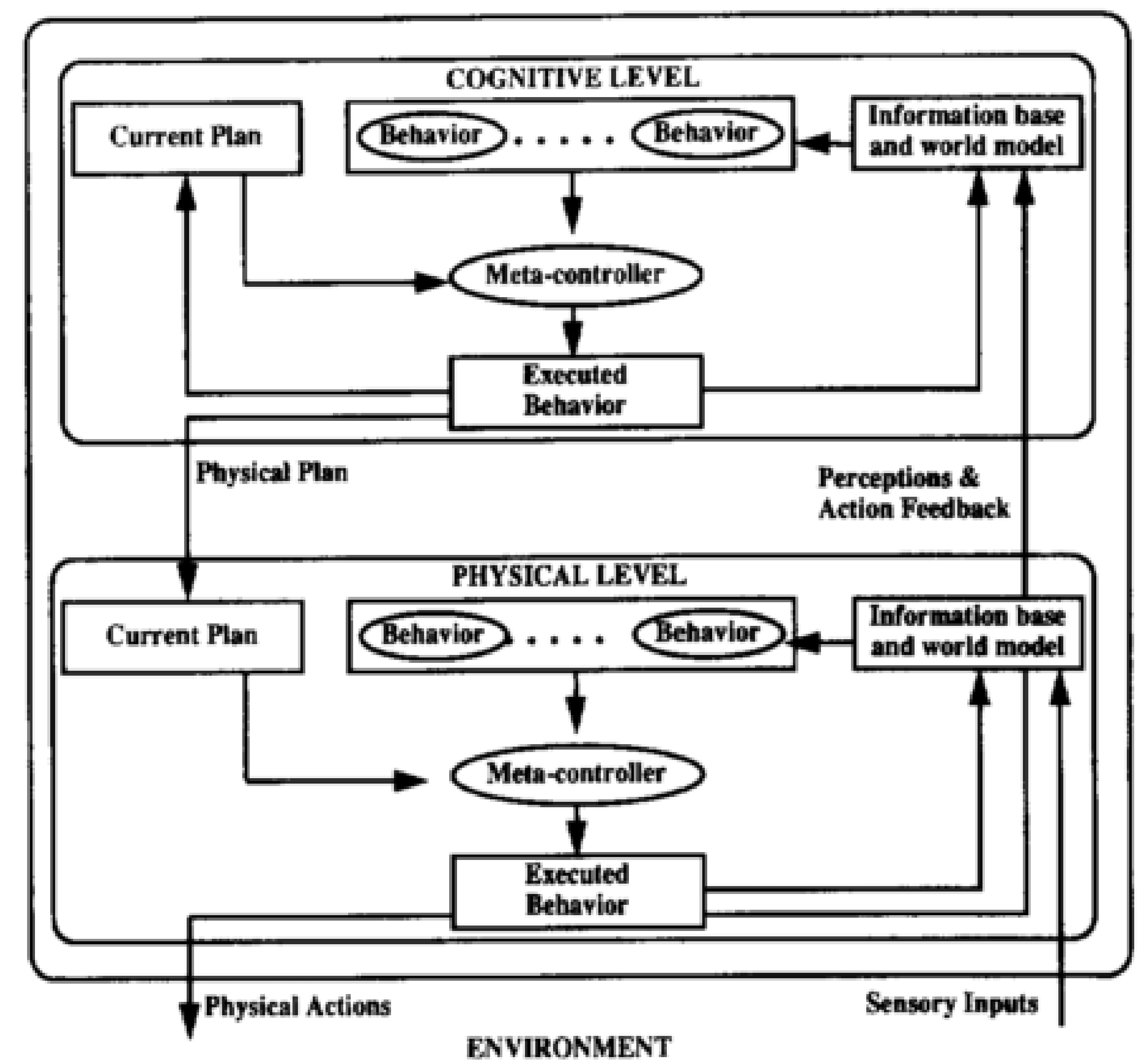
Managed system	IPM system

	Centralized	Decentralized	Distributed	Non-distributed
Control loop	x		x	

Two independent controls

Systems Mission	The IPM systems must perceive, reason, and act to achieve multiple behaviors in dynamic, uncertain, complex environments

GENERAL DESCRIPTION



Layer represent the physical level and the cognitive level of AIS system. The layers and the relations follow the pipe and filter style. Specifically,
AISRA uses a bidirectional pipe and filter style in which each level reads from two input data streams and writes to two output data streams. Each layer, itself, comprises a number of components, organized in a blackboard style, to allow for a range of potentially complex behavior.
Physical level layer is oriented to represent the context-awareness property of AIS systems. Physical level implement perception (e.g., through sensors) and action (e.g., through effectors) in the external environment. The Cognitive level layer aim to represent the situation-aware property. This layer implement more abstract reasoning activities such as situation assessment (e.g., analysis), planning, problem-solving (e.g., execution of plans). Information flow is bidirectional. The results of cognitive behaviors can influence physical behaviors and vice versa. Meta-controllers in each layer aim to select the best behavior to be executed.

Rationale

Authors report that the use of hierarchical layers and pipe and filter styles introduces advantages in modularity, which allow easy replacement or enhancement of individual levels and facilitate construction of complex behaviors and manipulation of higher level concepts. Moreover, such combination allow the concurrent execution of behaviors in both levels, improving their performance. Additionally, the use of the blackboard style to represent behaviors have allowed the flexible run-time behavior requirement of AIS systems.

Liabilities

Reference Architecture concept and objective
It prescribes specific topologies of components and interactions, and can be used to build a large set of different agents, each with its own capabilities and limitations, but all based on the same general framework (Garlan et al 1994). The RA gives an immediate understanding of a whole class of possible IPM systems. The RA's objective is to support the shared functional requirements of IPM agents and provides a framework for configuring diverse application-specific sets of components.

Other architectural decisions
Ontologies to represent information base and world model in both layers. Interoperability among application-specific components configured within an IPM agent, is achieved through the IPM shared ontology. Performance efficiency is addressed through considering cognitive and physical layers and their communication as pipes and filters. Dynamical adaptation of agents is addressed by events allocated into the blackboard and delivered to the connected behaviors. Behaviors have standard interfaces to interpret events and send results. Robustness and flexibility are addressed in the sense that agents can select the better behavior in runtime to achieve the agent's goal.

Requirements

Mapping to the MAPE-K model

Physical layer

Monitoring	Through pipes, it receives sensory inputs
Analysis	Behaviors are triggered by events (modifications in the IV/WM due to sensory inputs or previously executed behaviors). The system must choose the best behavior. For this, behaviors defines an standard interface.
Planning	Performed through control plans hosted in the IB/WM
Executing	Through meta-controller, which executes the most appropriate enabled behavior
Knowledge	Information Base/World Model, which houses a system's factual knowledge, descriptions of its potential behaviors, and a temporally organized representation of its run-time perception, reasoning, and action
Sensor	pipe transferring environment inputs
Actuator	Executed behavior (methods to achieve particular tasks)

Cognitive layer

Monitoring	Through pipes, it receives perceptions and action feedback from physical level (i.e., IB/WM at physical level))
Analysis	Behaviors are triggered by events (modifications in the IV/WM due to sensory inputs or previously executed behaviors). The system must choose the best behavior. For this, behaviors defines an standard interface.
Planning	Performed through control plans hosted in the IB/WM
Executing	Through meta-controller, which executes the most appropriate enabled behavior
Knowledge	Information Base/World Model, which houses a system's factual knowledge, descriptions of its potential behaviors, and a temporally organized representation of its run-time perception, reasoning, and action
Sensor	pipe transferring perceptions and actions feedback from the IB/WM at physical level
Actuator	Executed behavior (methods to achieve particular tasks), which sent through a pipe the new plan to the physical level.

DATA EXTRACTION

Institution	IBM
Author	IBM
Year	2006
Title	An architectural blueprint for autonomic computing

	ACM	IEEE	SCOPUS	WEB OF SCIENCE	COMPENDEX	SCIENCE DIRECT	SPRINGER	SNOWBALLING
Source								X

Venue	

	Conference	Journal	Book chapter	Report
Type of paper				X

Keywords						

Application domain	IT systems

	TRL1	TRL2	TRL3	TRL4	TRL5	TRL6	TRL7	TRL8	TRL9
TRL level									X

	General level		Major level				Primitive level		Others
Adaptive properties	Self-adaptive	Self-managed	Self-configuring	Self-healing	Self-protecting	Self-optimizing	Self-awareness	Context-awareness	
		x							

Quality attributes	interoperability								

Architectural patterns	Layers	Enterprise service bus pattern							
	Repository								

Industry-oriented (name of industry)	IBM

Structure adapted	Autonomic managers and human system administrators should interact with individual components by reading and setting high-level policies, rather than by reading and setting the thousands of configuration parameters on today's components.

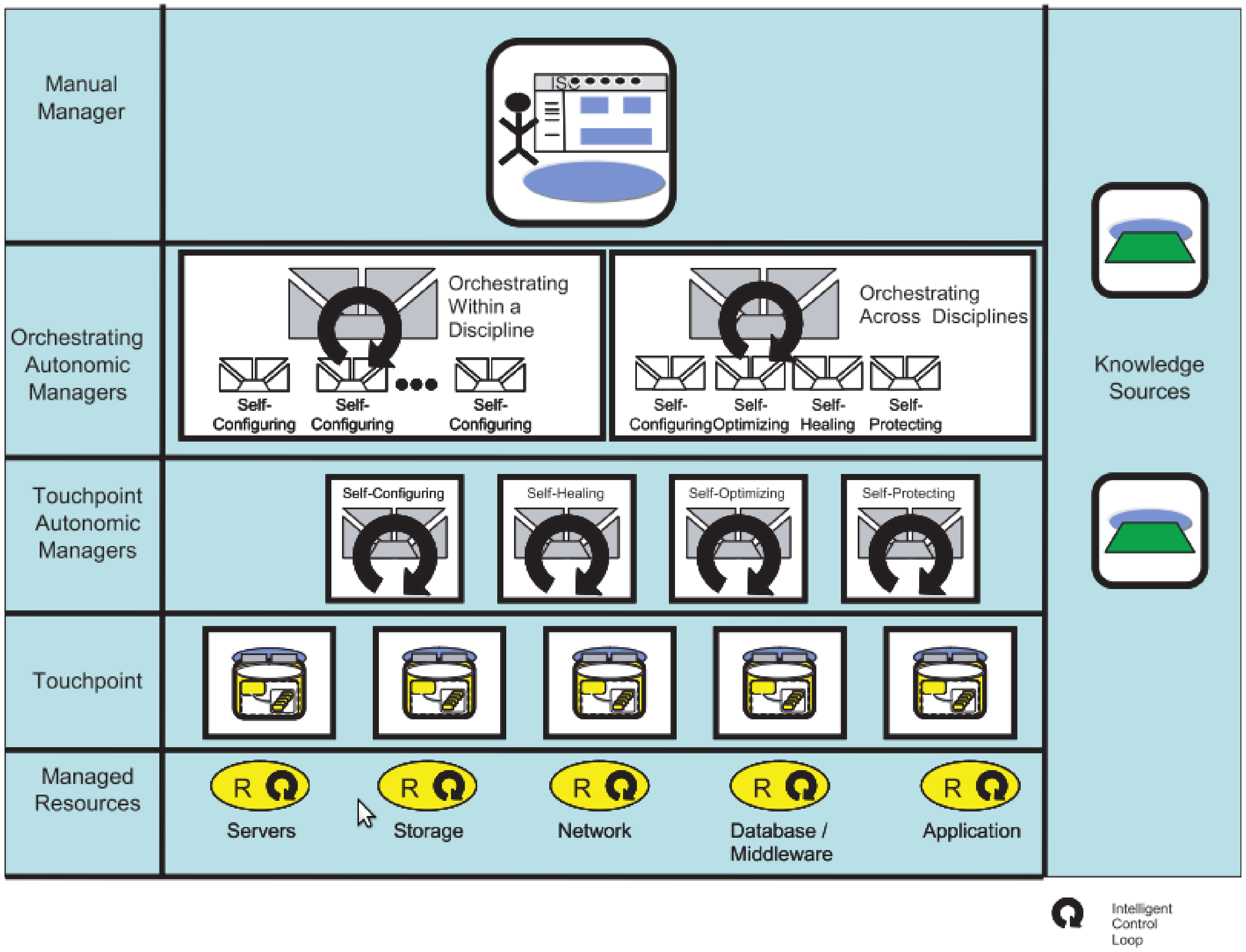
Managed system	Any managed resource and the managing system

	Centralized	Decentralized	Distributed	Non-distributed
Control loop		x	x	

Mission	IT systems must perform self-management autonomic capabilities to anticipate IT system requirements and minimize human intervention

Mission	IT systems must perform self-management autonomic capabilities to anticipate IT system requirements and minimize human intervention.

GENERAL DESCRIPTION



Liabilities

Reference Architecture concept

Other architectural decisions

Requirements

DATA EXTRACTION

Institution	Departament of computer science, Illinois Institute of Technology, Chicago, Illinois, USA
Author	Berhane Zewdie and C.R. Carlson
Year	2006
Title	Adaptive component paradigm for highly configurable business components

	ACM	IEEE	SCOPUS	WEB OF SCIENCE	COMPENDEX	SCIENCE DIRECT	SPRINGER	MANUAL
Source		x	x					

Venue	IEEE International Conference on Electro/information Technology

	Conference	Journal	Book chapter	Report
Type of paper	x			

Keywords	architecture	component	component specifi	design princip	adaptive system	

Application domain	business components - technology domain

	TRL1	TRL2	TRL3	TRL4	TRL5	TRL6	TRL7	TRL8	TRL9
TRL level	x								

	General level		Major level				Primitive level		Others
Adaptive properties	Self-adaptive	Self-managed	Self-configuring	Self-healing	Self-protecting	Self-optimizing	Self-awareness	Context-awareness	
			x						

Quality attributes	dependability	availability	Mean-time betwee	trhroughput	capacity	latency	safety	security	
	extensibility								

Architectural patterns	Layers-> three horizontal layers representing the ACP (Active component paradigm). There is a vertical layer that crosscut horizontal layers, representing the EFP (Extra functional properties)

Industry-oriented (name of industry)	none

Structure adapted	enterprise component

Managed system	components

	Centralized	Decentralized	Distributed	Non-distributed
Control loop	x			x

Mission	To change the behavior of the component without changing the code base (customization without programming) in runtime and dynamically

Mission	To change the behavior of the component without changing the code base (customization without programming) in runtime and dynamically

GENERAL DESCRIPTION

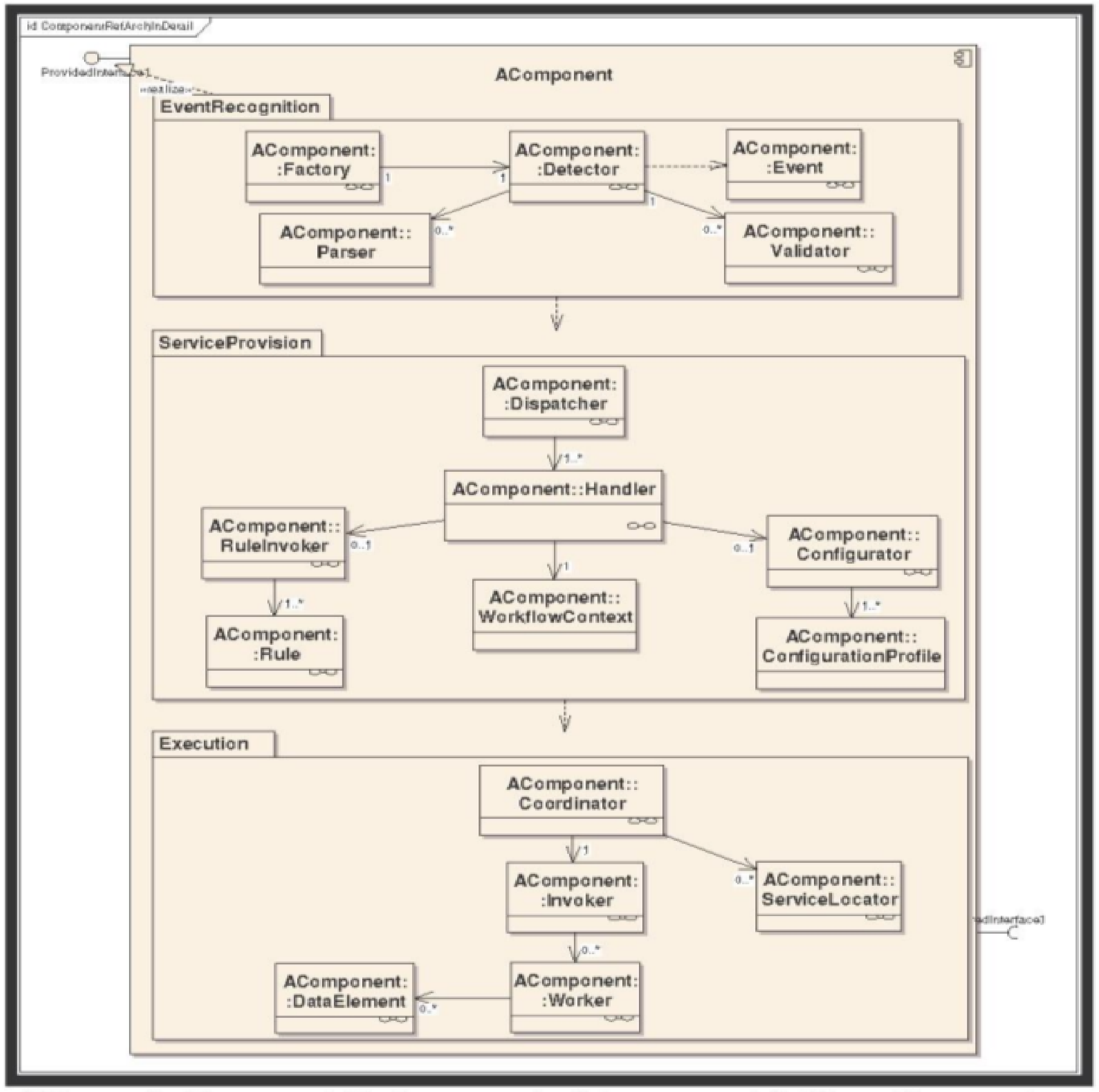


Figure 3: Proposed Reference Architecture with Key Abstractions

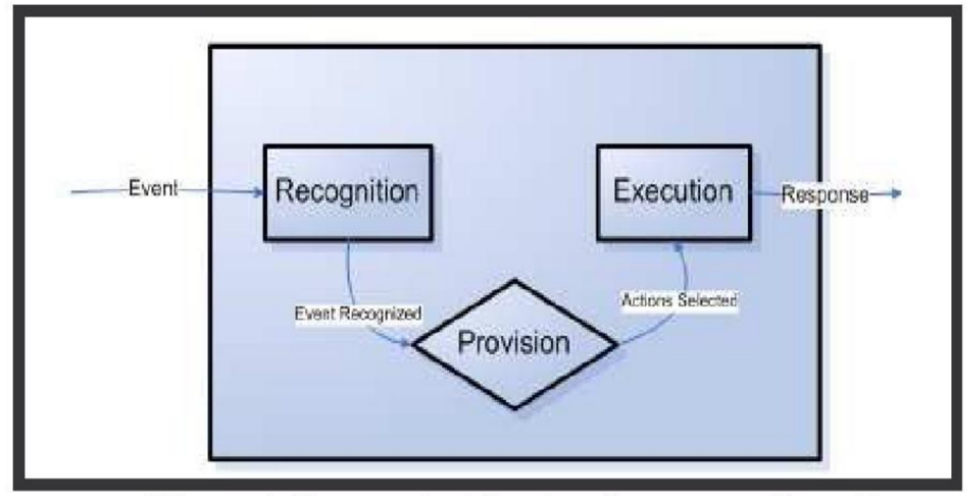


Figure 1: Proposed Adaptive Component Paradigm

Event -> Recognition -> Provision -> Execution -> *Response*

The Reference Architecture is based on the Adaptive Component Paradigm (ACP). The RA describes an adaptive component whose internal structure is composed by layers. Architectural layering helps to restructure applications that can be decomposed into groups which each group assumes a specific task at a particular level of abstraction. The RA is composed by three layers, which each layer corresponds to each phase of the ACP.

The Separation of Concerns principle was used to enforce independence and cooperation, because, this principle states that software should be decomposed in such a way that different concerns or aspects of the problem are solved in well separated modules of the software. Such principle allows address the following issues: representing business rules and processes, extra-functional property, and separating policy and execution.

Quality attributes are addressed by the Extra functional properties layer, a layer that crosscut all three ACP layers.

Extensibility is achieved by defining simple and comprehensive key abstractions with un-ambiguous functionality that can be implemented in a plug-in manner. Extensibility is the property of a technology that promotes evolution of modules without causing changes to other modules.

Processes and rules need to be configurable dynamically by decoupling their configuration from their execution and externalizing the configuration information. This approach facilitates the adaptability/extensibility of the component to changes in business demand and context of use, thereby conforming to the open-closed principle and cusstomization without programming.

Orthogonality is achieved by using the separation of concerns principle where isolation and, therefore, loosely coupled system are guarantee. Concerns related to policy decision and execution are separately handled as can be seen in the use of rule selection (policy selection) and rule execution and also the use of action composition (policy selection) and transaction (execution)

The policy issues can be best achieved by externalizing the decision making information, hence, achieving dynamically changing the component behavior without code changes.

Liabilities

Reference Architecture concept
Reference architecture provides architectural guidance that can be used to organize and standardize the development of software with the same constituent parts but in different context and application domains

Other architectural decisions

Requirements

Mapping to the MAPE-K model

Adaptive Component internal reference architecture

Monitoring	Event recognition Layer → Contains the Detector component, which is poised waiting for an event to happen. This component is used to recognize event occurrence in the form of state change, method execution, and message event arrival.
Analysis	Event recognition Layer → handles the responsibility of capturing requests and deciding whether the service request is relevant to the component or requires a response. This layer creates an instance of the component. Valid requests are passed to the Service provision layer.
Planning	Service provision Layer → Verifies various applicable state, context and business rules. Identifies and composes proper configuration actions with execution sequence. It sends selected actions to the execution layer.
Executing	Execution Layer → Responsible in executing the decision made in the service provision layer (i.e., execute rules that allow components reconfiguration)
Knowledge	Each layer has its own knowledge components. In the ERL the Parser component contains information to identify the nature of the event and it stores events for further use. In the SPL, the Configuration Profile component contains policy information used to configure component properties. Moreover, the Rule component stores business constrains, and the Workflow Context component contains information about actions to be executed. The EL has the Data Element component which holds business data.
Sensor	Any component that send requests to the Acomponent.
Actuator	Execution Layer → through the Worker component which is responsible for accomplishing the actual task to provide response for the event.

DATA EXTRACTION

Institution	**Karlsruhe Institute of Technology - Institute AIFB**
Author	Lei Liu, Stefan Thanheiser, Hartmut Schmeck
Year	2008
Title	A Reference Architecture for Self-organizing Service-Oriented Computing

	ACM	IEEE	SCOPUS	WEB OF SCIENCE	COMPENDEX	SCIENCE DIRECT	SPRINGER	MANUAL
Source							x	

Venue	ARCS 2008

	Conference	Journal	Book chapter	Report
Type of paper	x			

Keywords	Service-oriented Computing, Organic Computing, Self-organization, Reference Architecture, Reference Model.

Application domain	SOA based systems

	TRL1	TRL2	TRL3	TRL4	TRL5	TRL6	TRL7	TRL8	TRL9
TRL level	x								

	General level		Major level				Primitive level		Others
Adaptive properties	Self-adaptive	Self-managed	Self-configuring	Self-healing	Self-protecting	Self-optimizing	Self-awareness	Context-awareness	Self-organizing
		x							

Quality attributes	reusability	scalability	flexibility						
	interoperability	agility	robustness						

Architectural patterns	Observer controller	Layers							
	SOA	VSM							

Industry-oriented (name of industry)	no

Structure adapted	SOA elements located at SOA layer

Managed system	SOA-based system

	Centralized	Decentralized	Distributed	Non-distributed
Control		x	x	

A control loop is used based on the Observer Controller architecture

Mission	To allow self-organization to SOA-based systems

GENERAL DESCRIPTION

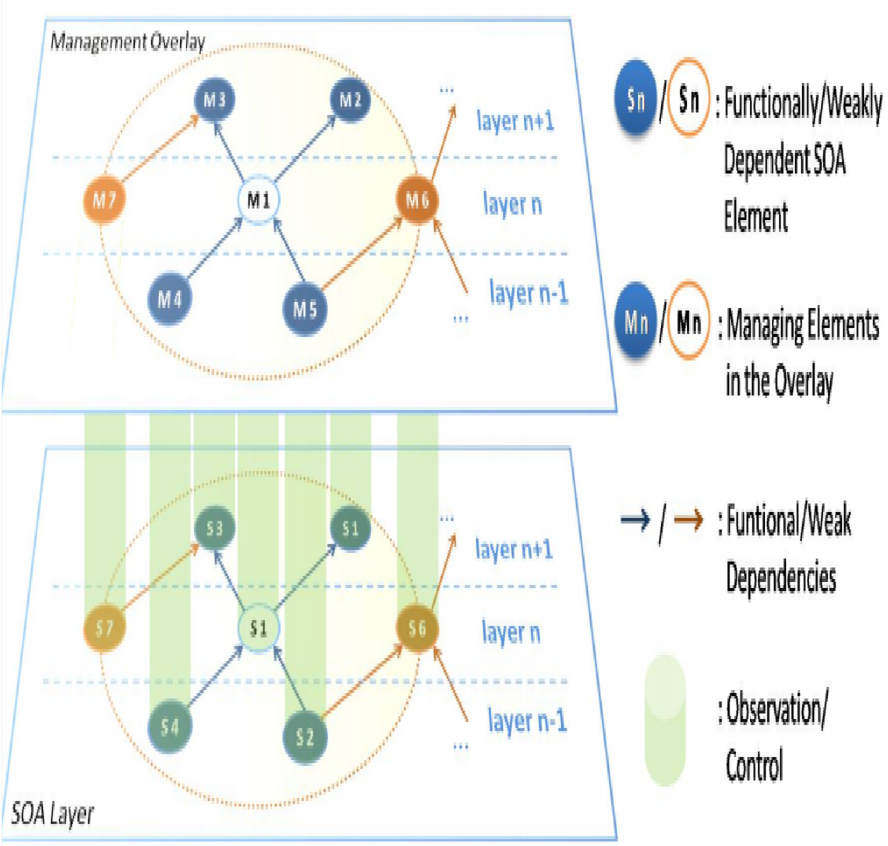


Fig. 2. Abstracted meta model for self-organizing SOA

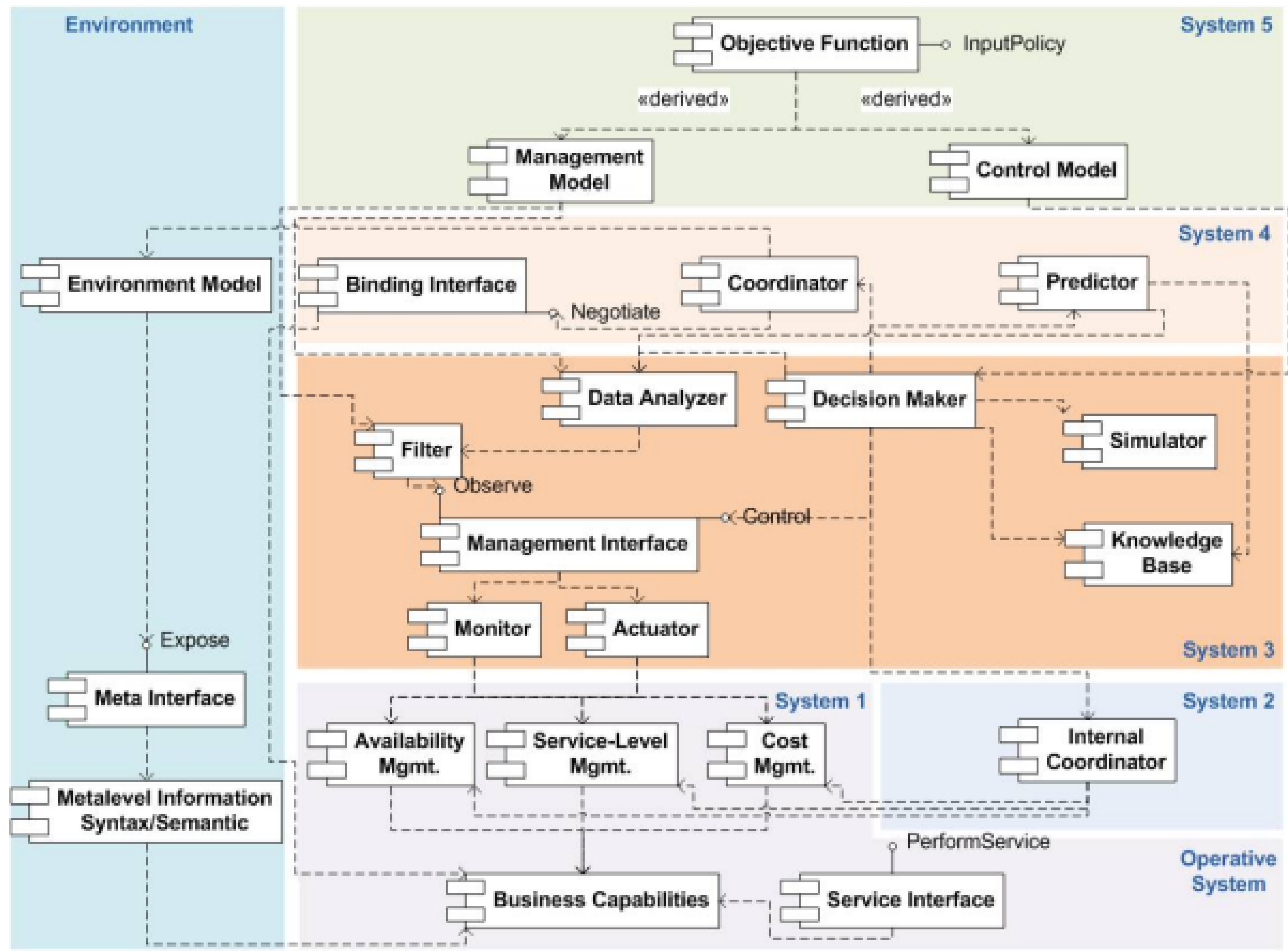


Fig. 3. Component view of a self-organizing component in OSOA with mapping to the VSM

Rationale

See sections 4.2 and 4.3

Liabilities

See Conclusions

Reference Architecture concept

A reference architecture serves as an architectural blueprint for constructing soft-
ware systems targeting particular problem domain(s) with specific functional,
behavioural, and quality attribute requirements [7]. It outlines a set of necessary
software components, their externally viewable interfaces as well as interrelation-
ships existing between them (e.g. data flows). [7]. Kazman, R., Clements, P., Bass, L.: Software Architecture in Practice. Addison-
WlRdi(2003)

Other architectural decisions

Requirements

Mapping to the MAPE-K model

Managing element reference architecture

Monitoring	monitor
Analysis	data analyser
Planning	predictor

Executing	controller
Knowledge	observation model and local data storing by each component
Sensor	SuOC sensor
Actuator	SuOC actuator

Based on the Observer Controller architecture (Richter2005)

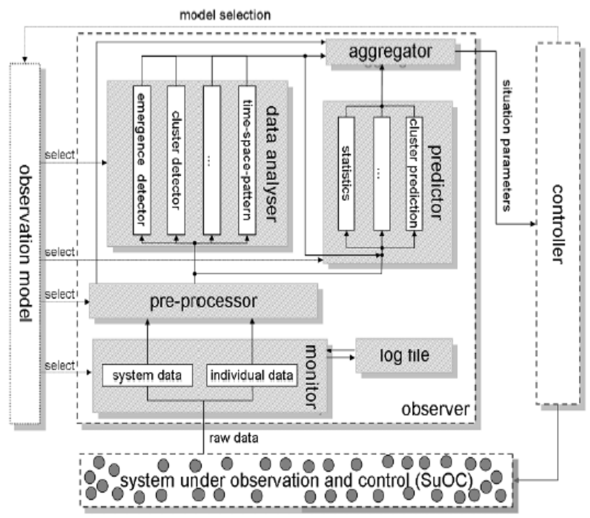


Figure 2: Generic observer architecture consisting of a monitor, a pre-processor, a data analyser, a predictor, and an aggregator.

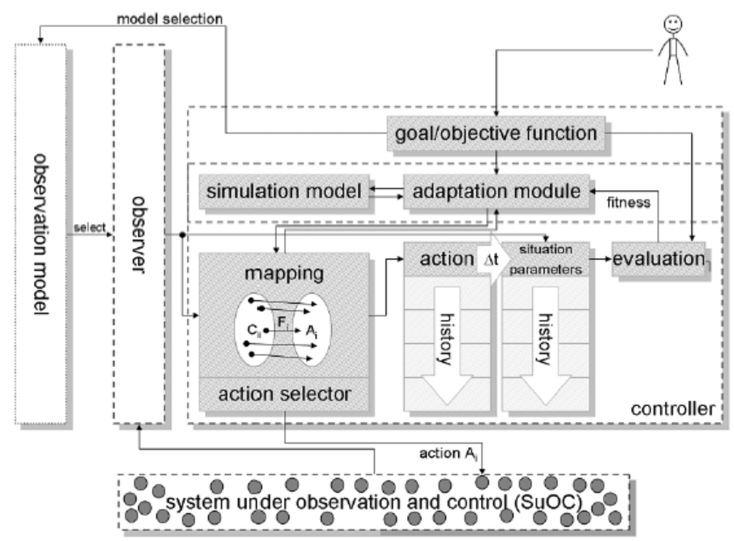


Figure 3: Generic controller architecture

DATA EXTRACTION

Institution	ICESI
Author	Bueno and Tamura
Year	2012
Title	A reference architecture for component-based self-adaptive software systems

	ACM	IEEE	SCOPUS	WEB OF SCIENCE	COMPENDEX	SCIENCE DIRECT	SPRINGER	MANUAL
Source								x

Venue	

	Conference	Journal	Book chapter	Report	Master thesis
Type of paper					x

Keywords						

Application domain	component-based self-adaptive systems

	TRL1	TRL2	TRL3	TRL4	TRL5	TRL6	TRL7	TRL8	TRL9
TRL level		x							

	General level		Major level				Primitive level		Others
Adaptive properties	Self-adaptive	Self-managed	Self-configuring	Self-healing	Self-protecting	Self-optimizing	Self-awareness	Context-awareness	
								x	

Quality attributes	throughput								

Architectural patterns	Layers								
	blackboard	pipe and filters							

Industry-oriented (name of industry)	no

Structure adapted	components behavior

Managed system	Component-based system

	Centralized	Decentralized	Distributed	Non-distributed
Control loop	x			x

Mission	Component-based self-adaptive systems must adapt their behavior depending on informations obtained from context

Mission	Component-based self-adaptive systems must adapt their behavior depending on informations obtained from context

GENERAL DESCRIPTION

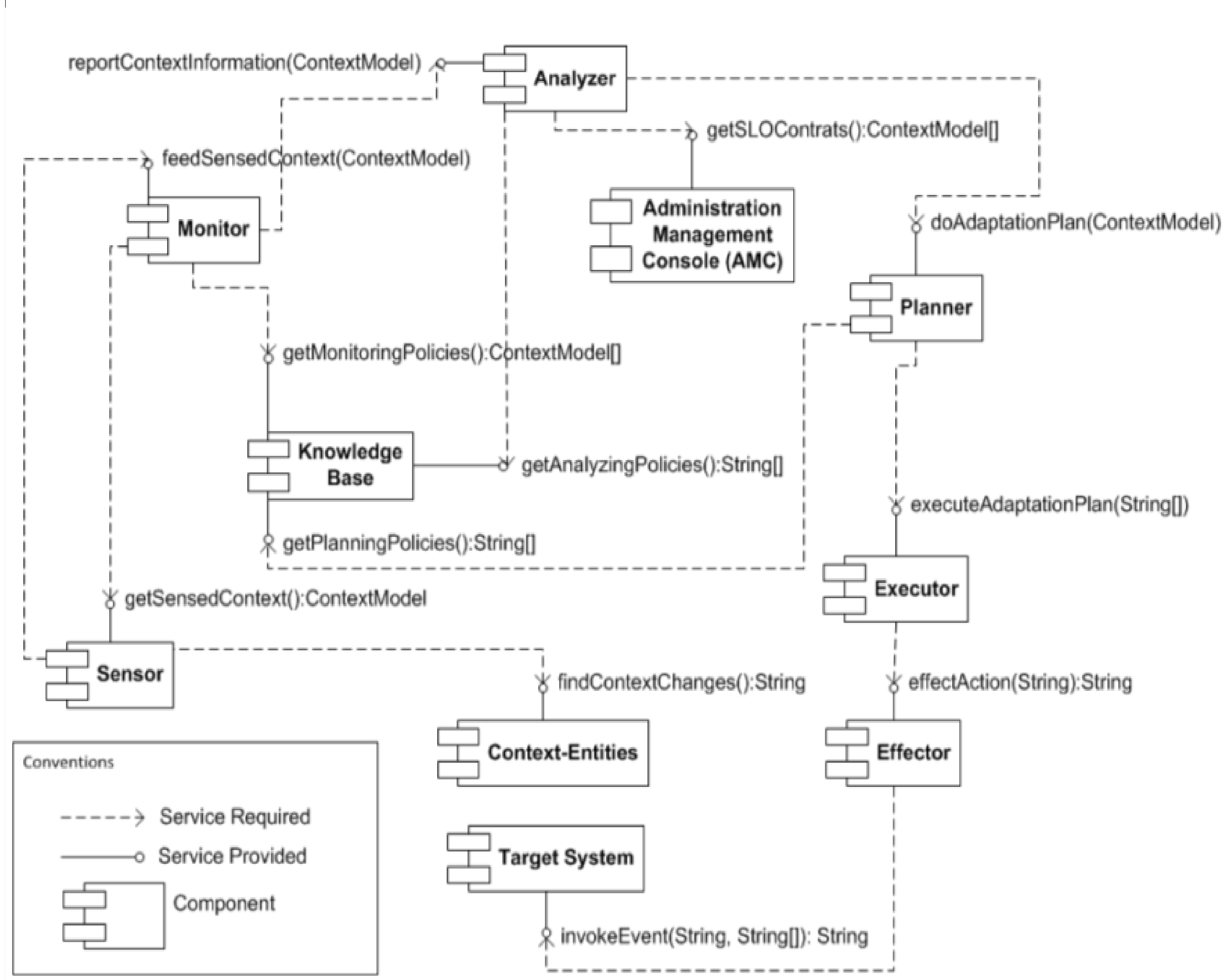


Figure 3.8: *Component-based reference architecture for self-adaptive systems*

Components implementing the MAPE loop are modeled as a filters and communication among such components following the pipe structure. The knowledge base is represented as a blackboard style.

Rationale

Liabilities

Reference Architecture concept

Other architectural decisions
It is based on the ACRA e DYNAMICO

Requirements

DATA EXTRACTION

Institution	UNESP
Author	Frank Jose Affonso and Elisa Yumi Nakagawa
Year	2013
Title	A reference architecture based on reflection for self-adaptive software

	ACM	IEEE	SCOPUS	WEB OF SCIENCE	COMPENDEX	SCIENCE DIRECT	SPRINGER	MANUAL
Source		x						

Venue	SBCARS

	Conference	Journal	Book chapter	Report
Type of paper	X			

Keywords						

Application domain	Reflective self-adaptive system

	TRL1	TRL2	TRL3	TRL4	TRL5	TRL6	TRL7	TRL8	TRL9
TRL level	x								

	General level		Major level				Primitive level		Others
Adaptive properties	Self-adaptive	Self-managed	Self-configuring	Self-healing	Self-protecting	Self-optimizing	Self-awareness	Context-awareness	reflectiveness
	x		x	x	x	x			

Quality attributes									

Architectural patterns	decomposition style								

Industry-oriented (name of industry)	none

Structure adapted	SaS

Managed system	Self-adaptive system

	Centralized	Decentralized	Distributed	Non-distributed
Control loop	x			x

Mission	To provide SaS with reflection, important resource for inspection and modification of the SaS at runtime. Such SaS presenting as main functionalities the

Mission	monitoring and adaptation at runtime without human intervention

GENERAL DESCRIPTION

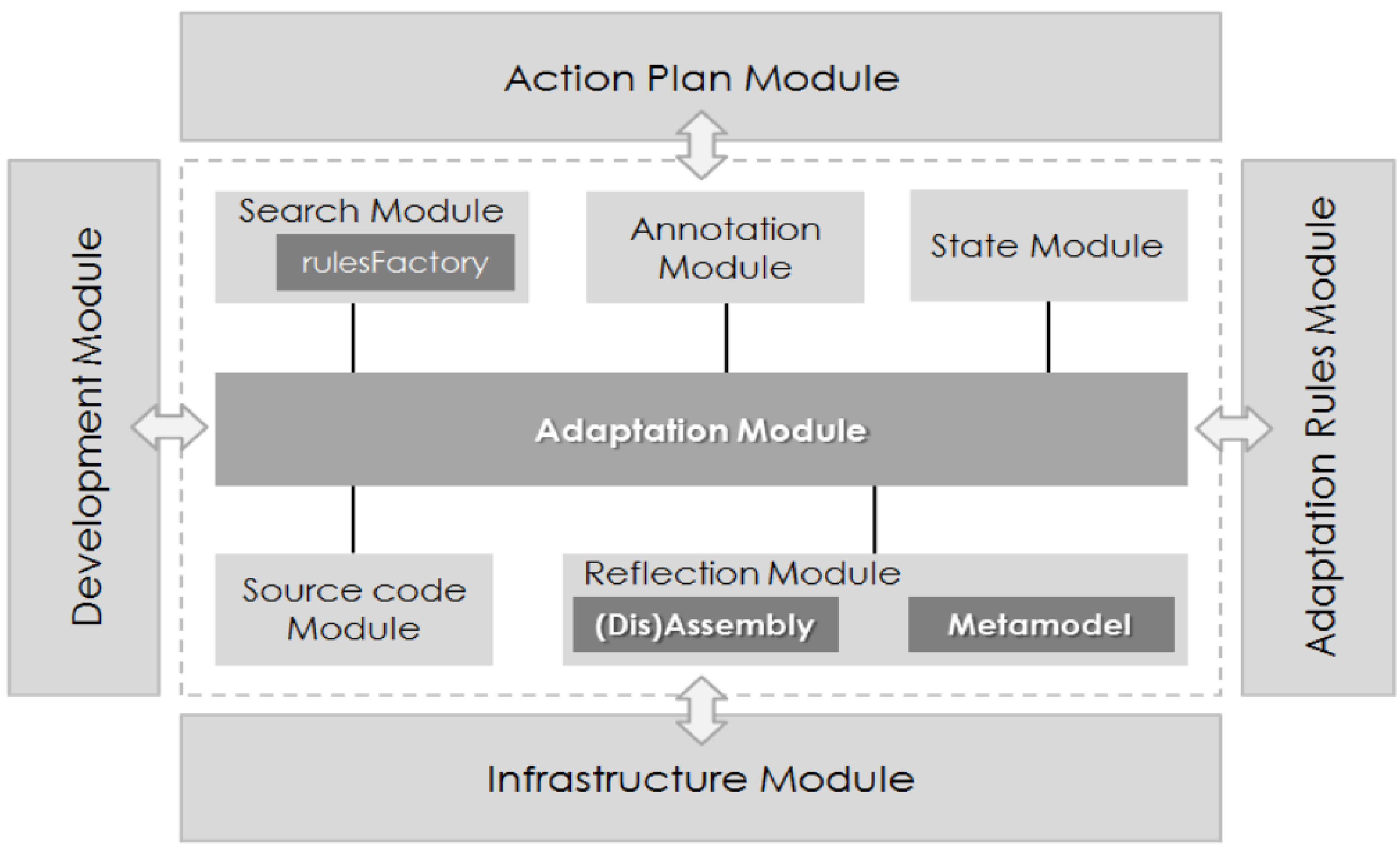


Figure 3. General representation of RA4SaS

Rationale

Liabilities

Reference Architecture concept

Other architectural decisions

Requirements

DATA EXTRACTION

Institution	Technical University of Madrid (UPM)
Author	Jennifer Perez, Jessica Diaz, Carlos Vidal, Daniel Rodriguez, Diego Fernandez
Year	2014
Title	Self-balancing distributed energy in power grids: an architecture based on autonomic computing

	ACM	IEEE	SCOPUS	WEB OF SCIENCE	COMPENDEX	SCIENCE DIRECT	SPRINGER	MANUAL
Source			x					

Venue	47th Hawaii international conference on system science

	Conference	Journal	Book chapter	Report
Type of paper	x			

Keywords						

Application domain	Smart Grids

	TRL1	TRL2	TRL3	TRL4	TRL5	TRL6	TRL7	TRL8	TRL9
TRL level									x

	General level		Major level				Primitive level		Others
Adaptive properties	Self-adaptive	Self-managed	Self-configuring	Self-healing	Self-protecting	Self-optimizing	Self-awareness	Context-awareness	
						x			

Quality attributes	Effectiveness	Performance effi	flexibility	deployability					
	reliability	scalability	interoperability						

Architectural patterns	layers								
	publish subscribe								

Industry-oriented (name of industry)	Implemented in real environments

Structure adapted	Policies to modify the Quantity of energy provided by microgrids/generating systems

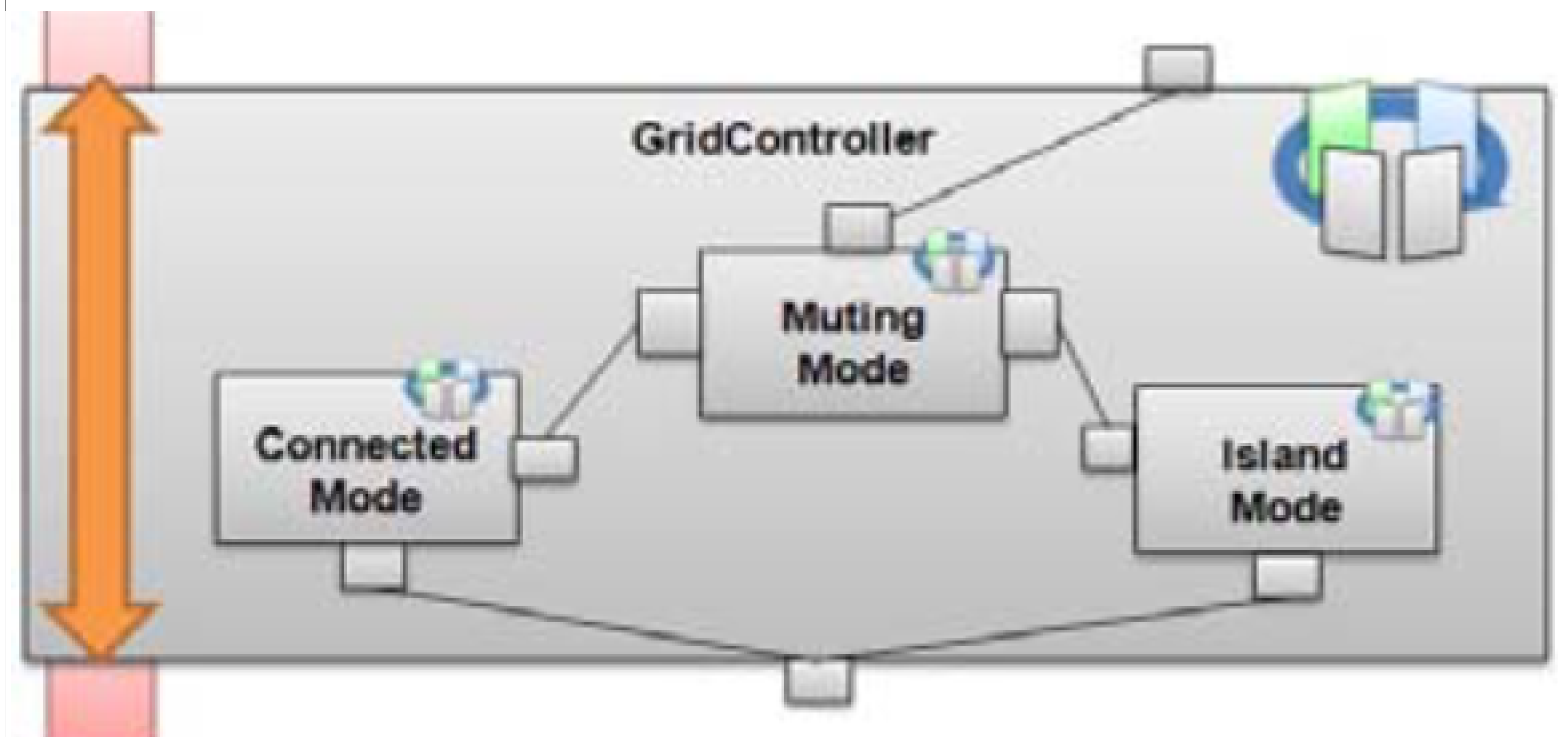
Managed system	Microgrid

	Centralized	Decentralized	Distributed	Non-distributed
Control loop	x		x	

Mission	Smart grids must perform load balance among generating systems to achieve maximum utilization and minimum response time

Mission	Smart grids must perform load balance among generating systems to achieve maximum utilization and minimum response time

GENERAL DESCRIPTION



Rationale

Liabilities

Reference Architecture concept

Other architectural decisions

Requirements

DATA EXTRACTION

Institution	Autonomous Systems Laboratory, UPM
Author	Ricardo Sanz, Carlos Hernandez, Julita Bermejo, Manuel Rodriguez, Ignacio Lopez
Year	2014
Title	Improved Resilience Controllers using congnitive patterns

	ACM	IEEE	SCOPUS	WEB OF SCIENCE	COMPENDEX	SCIENCE DIRECT	SPRINGER	snowballing
Source								x

Venue	19th World Congress The International Federation of Automatic Control (IFAC)

	Conference	Journal	Book chapter	Report
Type of paper	x			

Keywords	design patterns	controller archite	reconfiguration	Model-based	meta control	

Application domain	autonomous mobile robots

	TRL1	TRL2	TRL3	TRL4	TRL5	TRL6	TRL7	TRL8	TRL9
TRL level				x					

	General level		Major level				Primitive level		Others
Adaptive properties	Self-adaptive	Self-managed	Self-configuring	Self-healing	Self-protecting	Self-optimizing	Self-awareness	Context-awareness	
			x						

Quality attributes	Fault-tolerance								

TRLTR

Architectural patterns	Epistemic Control Loop Pattern		Deep model reflection pattern						
	Meta Control Pattern								

Industry-oriented (name of industry)	

Structure adapted	configuration of the robot's running control application

Managed system	robot's control application

	Centralized	Decentralized	Distributed	Non-distributed
Control loop	x			x

Mission	Autonomous mobile robots must handle any kind of uncertainty whether environmental or internal

Mission	Autonomous mobile robots must handle any kind of uncertainty whether environmental or internal

GENERAL DESCRIPTION

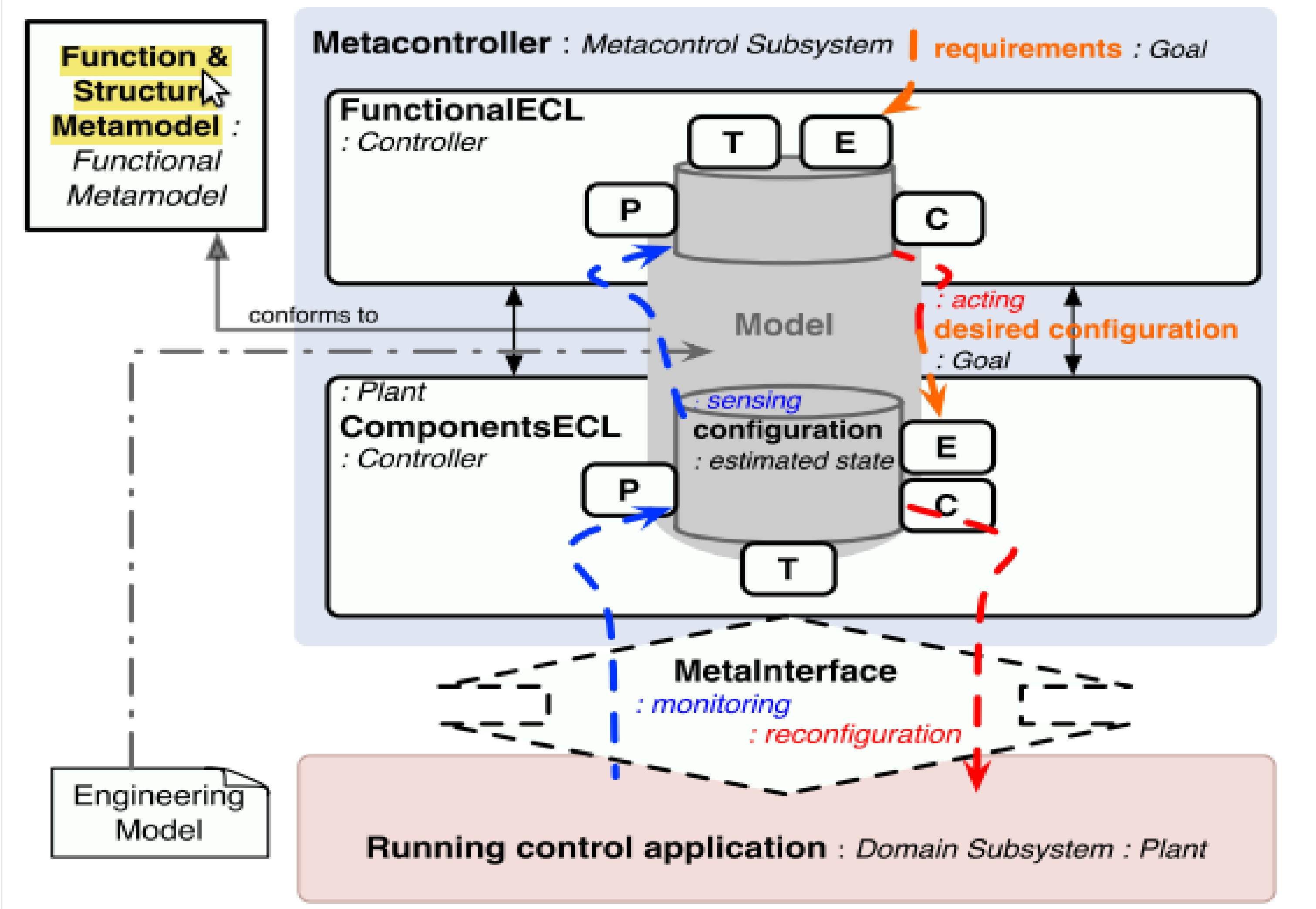


Fig. 4. The interplay of the main elements of the OM Ar-

Liabilities

Reference Architecture concept

Other architectural decisions

Requirements

DATA EXTRACTION

Institution	Computer science department, university of southern california
Author	Hossein Tajalli and Nenad Medvidovic
Year	2014
Title	IDARE-a reference architecture for integrated software environments

	ACM	IEEE	SCOPUS	WEB OF SCIENCE	COMPENDEX	SCIENCE DIRECT	SPRINGER	MANUAL
Source			x					

Venue	Software - Practice and Experience

	Conference	Journal	Book chapter	Report
Type of paper		x		

Keywords	reference archit	development env	Run-time environm	Self-adaptation		

Application domain	SALES

	TRL1	TRL2	TRL3	TRL4	TRL5	TRL6	TRL7	TRL8	TRL9
TRL level	x								

	General level		Major level				Primitive level		Others
	Self-adaptive	Self-managed	Self-configuring	Self-healing	Self-protecting	Self-optimizing	Self-awareness	Context-awareness	
Adaptive properties		x							

Quality attributes	adaptability	robustness							
	Fault-tolerance								

Architectural patterns									

Industry-oriented (name of industry)	

Structure adapted	Run-time applications

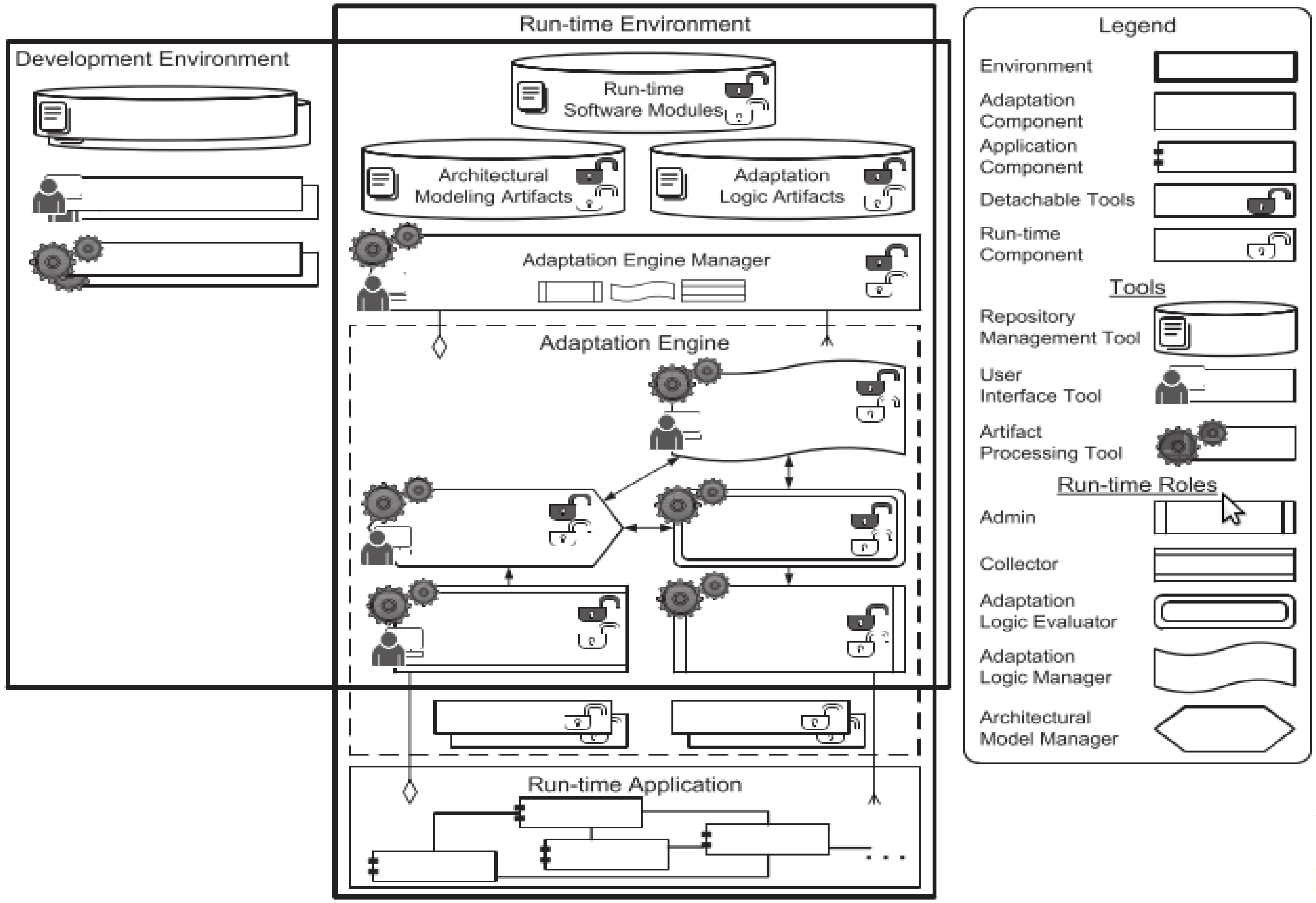
Managed system	adaptation engine

	Centralized	Decentralized	Distributed	Non-distributed
Control loop	x		x	

Mission	A SALE must provide development time and run-time tools to support all stages in the life-cycle of a self-adaptive software system. In particular, such

Mission	environment must enhance and automate software maintenance and adaptation stages, according with adaptations offered by developers.

GENERAL DESCRIPTION



Liabilities

Reference Architecture concept

Other architectural decisions

Requirements

DATA EXTRACTION

Institution	Technische Universitat Dresden, IRISA, SINTEF, and Fraunhofer IESE
Author	Uwe Abmann, Sebastian Gotz, Jean-Marc Jezequel, Brice Morin, and Mario Trapp
Year	2014
Title	A reference architecture and roadmap for models@runtime systems

	ACM	IEEE	SCOPUS	WEB OF SCIENCE	COMPENDEX	SCIENCE DIRECT	SPRINGER	MANUAL
Source							X	

Venue	Models@runtime, LNCS 8374, pp. 1-18

	Conference	Journal	Book chapter	Report
Type of paper			X	

Keywords						

Application domain	reflective systems, specifically, models@run.time systems. Important domains can be cyber-physical systems and safety-critical systems

	TRL1	TRL2	TRL3	TRL4	TRL5	TRL6	TRL7	TRL8	TRL9
TRL level	x								

	General level		Major level				Primitive level		Others
Adaptive properties	Self-adaptive	Self-managed	Self-configuring	Self-healing	Self-protecting	Self-optimizing	Self-awareness	Context-awareness	reflective properties
		X			X				

Quality attributes	tractability	decidability	flexibility						
	safety	reflectiveness	assurance						

Architectural patterns	three layers as proposed by kramer 2009

Industry-oriented (name of industry)	

Structure adapted	the managed system adapted by the models@run.time system, and the configuration management layer adapted by the goal management layer

Managed system	any observable and controllable system (e.g., robot, or a managing models@run.time system again)

	Centralized	Decentralized	Distributed	Non-distributed
Control loop	x			x

Mission	To enable unanticipated adaptations while ensuring safety

Mission	To enable unanticipated adaptations while ensuring safety

GENERAL DESCRIPTION

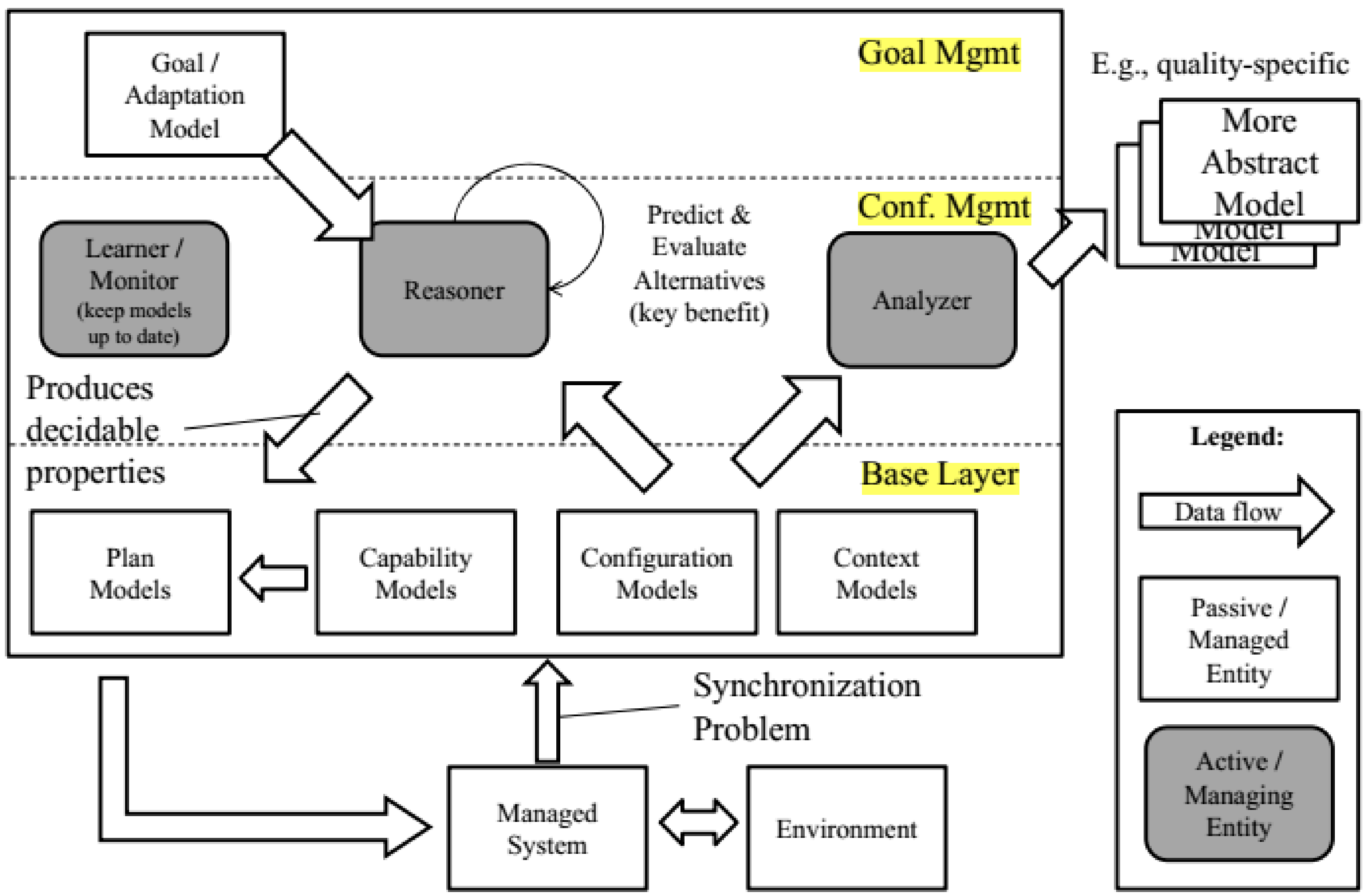


Fig. 1. Reference Architecture for models@run.time Systems

Rationale

To maintain tractability (easy to modify or adapt), the models@run.time systems abstract from certain aspects of their code, maintaining runtime models of themselves, which form the basis of reflection. Manageable reflection is addressed by using models specified in the base layer. This allow to reason about alternatives to reach systems goals and consequences of the particular actions through the reasoner and analyzer modules. In this context, safety levels can be maintained through selecting the best alternative. Layered structure allows that a models@run.time system can be managed by other models@run.time system, and therefore create complex systems addressing manageable reflection. The use of models allow predictive reflection, that is, the ability of reasoning about future configurations of the system ensuring desired properties such as safety. Moreover, models allow tractability by abstraction, that is, the ability of the analyzer to abstract the information used by the reasoner reducing the reasoning task's complexity and, thus, to get desidability and, finally, tractability of the overall system.

Liabilities

It is not applicable to distributed CPS or SCS. Improvements need to be made.

Reference Architecture concept
None

Other architectural decisions
They use the control loop but not follow the MAPE-K approach. Moreover, follow the three layer structure proposed by kramer. Managed and managing systems are loosely coupled, leaving to synchronization issues.

Requirements

DATA EXTRACTION

Institution	Departamento de Computación, FCEN, Universidad de Buenos Aires, Argentina
Author	Braberman, V., D'Ippolito, N., Kramer, J., Sykes, D., Uchitel, S.
Year	2015
Title	MORPH: A reference architecture for configuration and behaviour self-adaptation

	ACM	IEEE	SCOPUS	WEB OF SCIENCE	COMPENDEX	SCIENCE DIRECT	SPRINGER	MANUAL
Source	x							

Venue	CISE'15

	Conference	Journal	Book chapter	Report
Type of paper	X			

Keywords	Self-adaptive systems	software architefcture		

Application domain	Any self-adaptive system requiring independent yet coordinated behaviour and configuration adaptation

	TRL1	TRL2	TRL3	TRL4	TRL5	TRL6	TRL7	TRL8	TRL9
TRL level		x							

	General level		Major level				Primitive level		Others
Adaptive properties	Self-adaptive	Self-managed	Self-configuring	Self-healing	Self-protecting	Self-optimizing	Self-awareness	Context-awareness	behavior adaptation
		x	x						

Quality attributes	coordinated adaptation	independent adaptation
	transparent adaptation	adaptation of system configuration and behaviour

Architectural patterns	Layers			**Others architectural decisions**	MAPE-k by layer	Rainbow framework to model the target system
					Master-slave decision pattern to provide coordination within layers elements	

Industry-oriented (name of industry)	none

Structure adapted	It is adapted the configuration and behavior of the target system.

Managed system	Target system following the rainbow framework

	Centralized	Decentralized	Distributed	Non-distributed
Control loop		x	x	x

The nature of target system is encapsulated. This RA is independent of components architecture

Mission	To allow configuration and behavior adaptation in runtime independently

Mission	To allow configuration and behavior adaptation in runtime independently

GENERAL DESCRIPTION

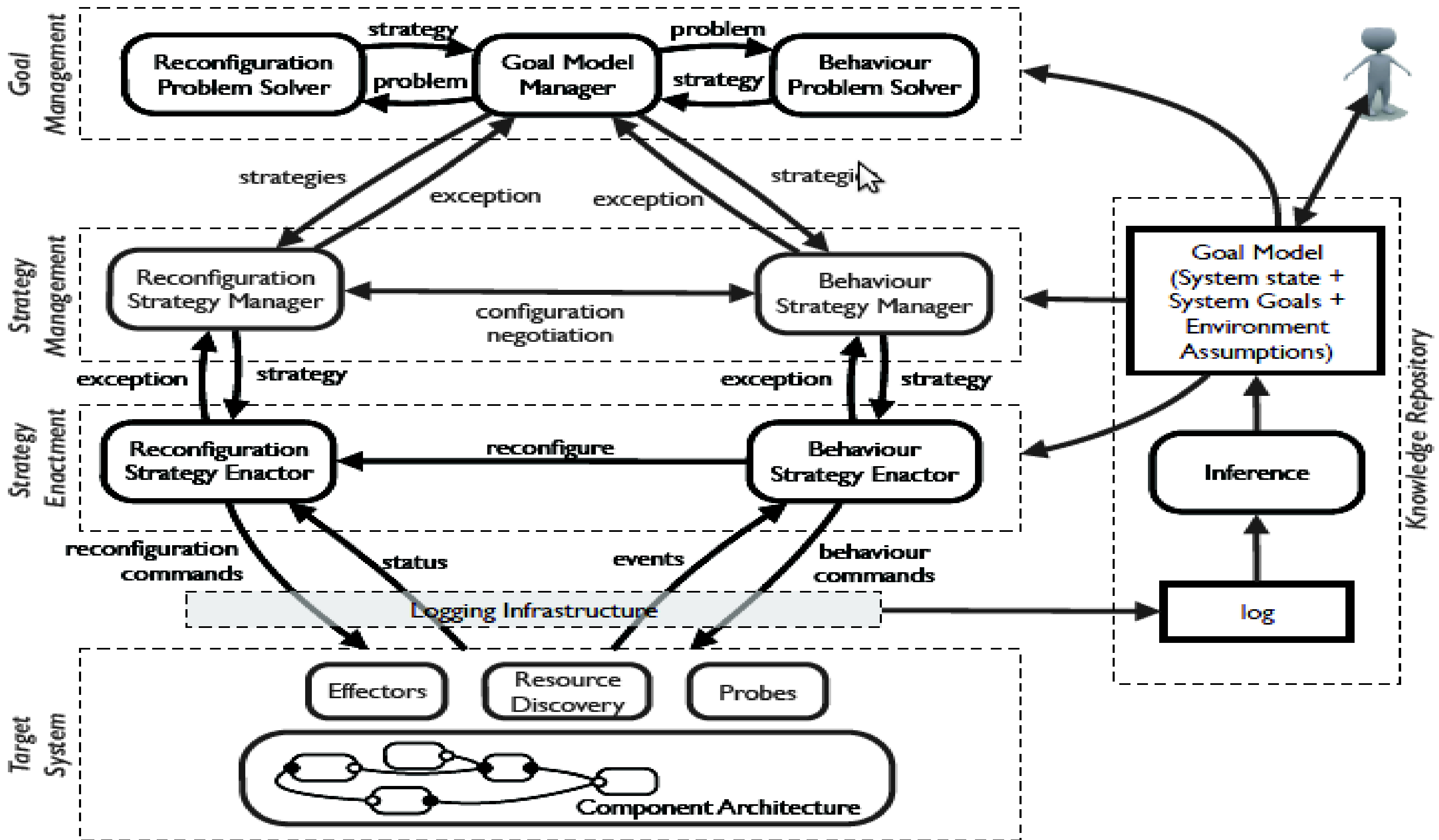


Figure 1: The MORPH Reference Architecture.

Rationale

It is presented by all element of the RA

Liabilities

Reference Architecture concept

Other architectural decisions

Requirements

Mapping to the MAPE-K model

Target system

Monitoring	Monitoring of the component architecture can be provided by PROBES that reveal state information. Monitoring information can be classified as information regarding components status (active, inactive, connected, killed), or events that flows from the target system to the strategy enactment layer.
Analysis	
Planning	

Executing	
Knowledge	
Sensor	
Actuator	effectors: API to configure components (add, remove and bind components) and API to invoking functional services (behaviour actions)

Strategic enactor layer

Monitoring	
Analysis	
Planning	
Executing	
Knowledge	
Sensor	
Actuator	

Strategic management layer

Monitoring	
Analysis	
Planning	
Executing	
Knowledge	
Sensor	
Actuator	

Goal management layer

Monitoring	
Analysis	
Planning	
Executing	
Knowledge	
Sensor	
Actuator	

DATA EXTRACTION

Institution	UFRJ
Author	Jesus M.T. Portocarrero, Flavia C. Delicato, Paulo F. Pires, Bruno Costa, Wei Li, Weisheng Si, Albert Y. Zomaya
Year	2017
Title	RAMSES: A new reference architecture for self-adaptive middleware in Wireless Sensor Networks

	ACM	IEEE	SCOPUS	WEB OF SCIENCE	COMPENDEX	SCIENCE DIRECT	SPRINGER	MANUAL
Source	x		x					

Venue	Ad Hoc Networks

	Conference	Journal	Book chapter	Report
Type of paper		X		

Keywords	autonomic comp	self adaptive sy	WSN	reference arch	Pi-adl	

Application domain	WSN

	TRL1	TRL2	TRL3	TRL4	TRL5	TRL6	TRL7	TRL8	TRL9
TRL level			x						

	General level		Major level				Primitive level		Others
Adaptive properties	Self-adaptive	Self-managed	Self-configuring	Self-healing	Self-protecting	Self-optimizing	Self-awareness	Context-awareness	
			x						

Quality attributes	resource utilization		interoperability						
	flexiblility		Fault-tolerance						

Architectural patterns	SOA	Layers		decorator pattern		aggregator escalator peer architectural style			
	broker pattern	Service components for SaS		aspect p2p architectural style					

Industry-oriented (name of industry)	

Structure adapted	nodes configuration

Managed system	WSN

	Centralized	Decentralized	Distributed	Non-distributed
Control loop	x		x	

Mission	To provide autonomic behavior to WSN

Mission	To provide autonomic behavior to WSN

GENERAL DESCRIPTION

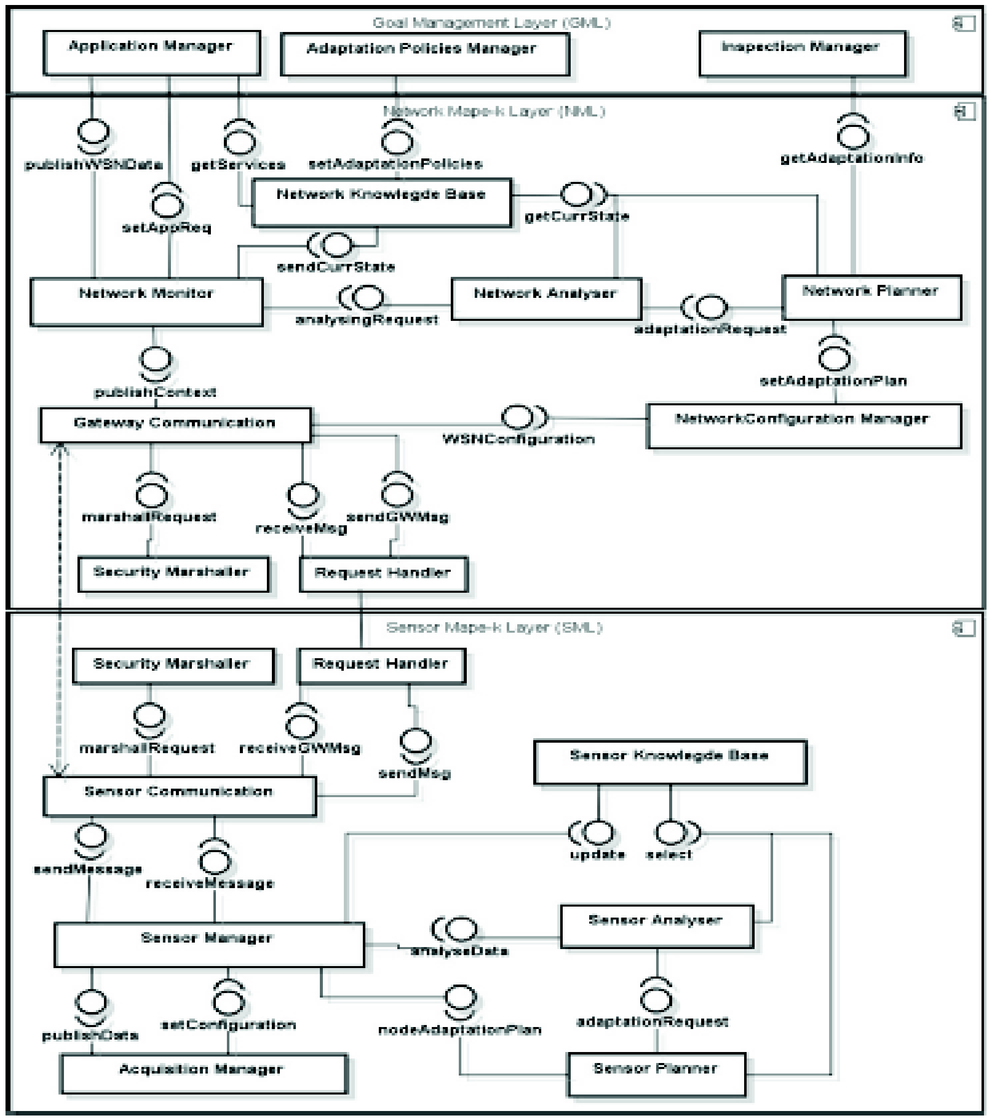


Fig 1. RA of a self-adaptive middleware for WSN

onale

Liabilities

Reference Architecture concept

Other architectural decisions

Requirements

DATA EXTRACTION

Institution	
Author	
Year	
Title	

	ACM	IEEE	SCOPUS	WEB OF SCIENCE	COMPENDEX	SCIENCE DIRECT	SPRINGER	MANUAL
Source								

Venue	

	Conference	Journal	Book chapter	Report
Type of paper				

Keywords						

Application domain	

	TRL1	TRL2	TRL3	TRL4	TRL5	TRL6	TRL7	TRL8	TRL9
TRL level									

	General level		Major level				Primitive level		Others
Adaptive properties	Self-adaptive	Self-managed	Self-configuring	Self-healing	Self-protecting	Self-optimizing	Self-awareness	Context-awareness	

Quality attributes									

Architectural patterns									

Industry-oriented (name of industry)	

Structure adapted	

Managed system	

	Centralized	Decentralized	Distributed	Non-distributed
Control loop				

Mission	

Mission	

GENERAL DESCRIPTION

Rationale

Liabilities

Reference Architecture concept