Generic Autonomic Adapter Architecture and Policy Model For Semantic Socio-Cyber-Physical Collaborative Network

Nazmul Hussain

Engineering and Applied Science
Aston University, Birmingham, UK

This dissertation is submitted for the degree of
Doctor of Philosophy

27 September 2020

This dissertation is affectionately dedicated to my lovely parents, who brought and grew me up in this world and prayed for me every day, encouraged me to continue my studies and helped me migrate to the UK. It is also dedicated to my beautiful little kids Tanzila Ziha Dulari, Hafsa Hussain and Hamza Hussain and my brother Dr Akmol Hussain who encouraged me and gave me financial support time to time as I needed.

Declaration

I hereby declare that this thesis presented for the degree of PhD has been composed solely by myself, the contents are original of my research and that this work has not been submitted, in whole or in part for any other degree or processional qualification except as specified in the text and Acknowledgements.

Nazmul Hussain
27 September 2020

Acknowledgements

"Allahumdulillah!", I am grateful to the most merciful and creator Almighty Allah (swt) for all the things that He blessed me with throughout my whole PhD journey and without His blessings, I would not be able to finish my PhD.

I would like to express my sincere gratitude to my supervisor, Dr Hai H. Wang for his endless patience, great motivation, excellent guidance and a tremendous amount of support throughout my PhD study. I am thankful to him for giving me the opportunity and freedom to pursue a PhD and was able to finish it under his supervision.

I am also very grateful to my associate supervisor, Dr Christopher D. Buckingham who was always more than generous with his expertise and time and helped me with his constructive feedbacks to improve this work. I am greatly indebted to him for partial funding of my PhD from the GRiST project and helping me to validate the work based on the GRiST case study. I am also very grateful to the other GRiST project members for supporting this thesis.

I would like to thank my colleague Dr Muhudin A. Mohamed and Dar Sanobar for their support and advice during the writing of this thesis.

Finally, and most importantly, I would like to thank my parents, brothers and sisters for their unconditional love and encouragement. I owe everything that I have achieved to them.

This dissertation work is supported partially by the Institute of Innovation and Technology (EIT) Health Knowledge Innovation Community (KIC) of GRiST project.

Abstract

The cyber-physical system aims to improve the quality of life of citizens by providing intelligent and automated services in a wide variety of sectors like transportations, healthcare, enterprises, self-driving cars, energy sectors and so forth. Recently, considerable amounts of researches have focused on integrating cyber-physical systems in a social context. The idea is to socially connect cyber-physical resources (i.e., physical devices, software elements, networked components, digital contents, etc.) so that they can interact and collaborative for autonomous decision making like humans social networking. However, several challenges remain concerning the designing appropriate methodologies, frameworks and techniques for supporting cyber-physical relation and collaboration within the social context. Most of the existing social software modelling focuses on maintaining human-to-human or human-to-object centric interaction only. Existing systems do not recognise how socio-cyber-physical resources can maintain their social status, communicate and interact with both humans and nonhuman entities. The reason may be the lack of understanding and limited approaches or methodologies to semantically (a formal characterisation of the information) represent the socio-cyber-physical resources relation and interactions in a collaborative network. This limits data integration, interoperability, and knowledge discovery from its underlying data sources. Semantic Web's ontology with a software agent model can help to overcome this limitation by describing and interconnecting socio-cyber-physical objects in a social space. The software agents can act as a representative of these resources to track, manage and update their collaborative activities in a social world.

Nevertheless, due to the exponential network growth and uncertainties, the states and relations among socio-cyber-physical objects may keep changing when they are in different situations. Therefore, it is an ardours task and error-prone for humans or traditional software agents to keep track, manage and maintain the larger number of socio-cyber-physical resources and their social dynamics. One potential and flexible solution to this problem is to leverage the autonomic computing approach with social and adaptive goals to make the socio-cyber-physical network self-managed and adaptive. Autonomic Computing (AC) approach has laid the necessary foundation to tackle this challenge by developing policy-based *Autonomic Adapter (AA)* model (e.g., autonomous agent). The AAs can continuously monitor socio-cyber-physical resource status, analyse the situation and make a collaborative decision based on the policy knowledge defined by the system administrator.

However, autonomic computing model must rely on input knowledge to decide self-management operations such as “what”, “where” and “how” to perform the adaptation to the system. Previously, adaptation approaches in a different context have been done in an ad-hoc manner based on the algorithms to predict future circumstances and embed in the program code. This approach is inflexible to dynamic and uncertain environments where system configuration needs to adjust frequently. Defining a flexible policy model and integrating policy into knowledge repository outside the code itself is the most appropriate to manage the autonomic system behaviours during the run-time. Sadly, there has been relatively a little work on developing appropriate policy model and specification language for domain neutral autonomic system.

To fulfil the above gaps, our proposed solutions in this thesis has three core contribution to the knowledge. First, we address the establishment of both socio-cyber-physical and human relations and interactions within a social-collaborative network. To achieve this, we propose a software agent-centric Semantic Social-Collaborative Network (SSCN) that provides the functionality to represent and manage cyber-physical resources in a social network. We discuss how nonhuman resources can be represented as socially connected nodes and manage by the software agents. The SSCN is supported by an extended ontology model for semantically describing the concept, properties and relations of human and nonhuman resources. A Java-based software agent API has been implemented to demonstrate some actions performed on behalf of the nonhuman resources in a real-world collaborative healthcare system called, GRiST (www.egrist.org). Second, we propose a Generic Autonomic Social-Collaborative Framework (GASCF) with a policy-based Autonomic Adapter (AA) architecture. The AAs are capable of monitoring system resources, analysing context information, and act accordingly using high-level policy. The AAs can also communicate and exchange data with other AAs through a social network for collaborative decisions making like human social interaction. Third, we propose Event-Condition-Action (ECA) rule-based policy model and specification language for AA by defining Policy Schema Definition (PSD) and Policy Script Specification (PSS) languages, modelled with XML syntax. Finally, we test and evaluate our approach by implementing it to the extended GRiST socio-healthcare service context and eGRiST clinical decision support system. We demonstrate and evaluate how socio-cyber-physical relation, interaction and autonomous decision-making is achieved by integrating AAs and using policy specification to manage AAs behaviour within socio-cyber-physical medical context.

Table of contents

List of figures

List of tables

Listings

List of Abbreviations

AA	Autonomic Adapter
AC	Autonomic Computing
CDSS	Clinical Decision Support Systems
CPS	Cyber-Physical System
CPSS	Cyber-Physical Social System
CSCW	Computer-Supported Cooperative Work
CW	Collaboration Workspace
CWS	Collaborative Work Systems
DC	Dublin Core
DTD	Document Type Definition
ECA	Event-Condition-Action
EIT	European Institute of Innovation and Technology
ESCN	Extended Social-Collaborative Network
FOAF	Friend Of Friend
GRaCE	Galatea Risk and Care Environment
GRiST	Galatea Risk and Safety Tool
GASCF	Generic Autonomic Social-Collaborative Framework
ICT	Information and Communication Technology
IT	Information Technology
IoT	Internet of Things
KIC	Knowledge Innovation Community
KR	knowledge Representation

MAPE-K	Monitor-Analysis-Plan-Execute and Knowledge
NHS	National Health Service
NFV	Network Function Virtualisation
OSM	Ontology Service Module
OWL	Web Ontology Language
PSD	Policy Schema Definition
PSL	Policy Specification Language
PSS	Policy Specification Script
RDF	Resource Description Framework
RSS	Really Simple Syndication
SC	Social Computing
SCIMS	Social Context Information Management System
SECO	Software Ecosystem
SDN	Software-Defined Networking
SGML	Standard Generalized Markup Language
SN	Social Network
SNA	Social Network Analysis
SIOC	Semantically Interlinked Online Communities
SSCN	Semantic Social-Collaborative Network
SPARQL	SPARQL Protocol and RDF Query Language
SW	Semantic Web
SWT	Semantic Web Technology
SWRL	Semantic Web Rule Language
QoS	Quality of Service
XML	Extensive Markup Language
XSD	XML Schema Definition

Definition of Terms

Autonomic Adapter	An intelligent agent model that can monitor and manage socio-cyber-physical resources using policy. They can also cooperate and communicate with each other through a social network and make collaborative decision and actions.
Autonomic Behaviour	Autonomic behaviour is characterised by self-configuration and self-healing capabilities, aimed at permitting the system to manage the failure of one or more of its agents and ensure continuous functioning.
Autonomic Computing	Autonomic Computing (AC) is a generic concept introduced by IBM inspiring from the human autonomic nervous system that brings together many fields of computing. Autonomic computing deal with complexity and uncertainties of the IT system, and aims at realising computing systems and applications capable of managing themselves without human intervention.
Agent	An agent is defined as an entity placed in an IT system, senses different event parameters and reacts to the changes in its environment based on reasoning process and apply the changes through actuator.
Artificial Intelligence (AI)	AI concerns the study and design of intelligent machines or software that are capable of performing a similar task of using computers to understand human intelligence.
Artifacts	The artifacts in a wide sense, a broad class of instruments, objects and tools that are used by agents in the distributed environment to configure and facilitate group decision-making, thinking and communication.

Cyber-Physical System	Cyber-physical system (CPS) is a large-scale and networked system, which consists of physical and computational elements.
Cyber-Physical-Social System	Cyber-Physical-Social systems (CPSSs) are the extension of Cyber-Physical systems (CPS), which seamlessly integrate cyber space, physical space and social space.
Cyber-Physical-Social Network	A third-generation social networks that take both the physical context as well as the virtual context of users into account.
Description Logic	Description logics (DL) is a knowledge representation language to represent the knowledge of an application domain in a structured and formally well-understood way. DLs differ from their predecessors, such as semantic networks and frames, in that they are expressive and equipped with a formal, logic-based semantics.
Data Integration	Data integration is the combination of technical and business processes used to combine data from disparate sources into meaningful and valuable information.
ECA Rule	The ECA (Event-Condition-Action) rule-based policy enforcement dictates on an occurred event, if condition satisfied, then perform action.
GRiST	GRiST (Galatean Risk and Safety Technology) is a web-based mental-health risk and safety management decision support system, which was a project sponsored by the European Union within its European Institute of Innovation and Technology Health Knowledge Innovation Community (EIT Health), which is currently used within the English National Health Service (NHS) and other organisations.
GASCF	The Generic Autonomic Social-Collaborative Framework (GASCF) is a high-level conceptual framework that is composed of multiple AAs that can collaborate and communicate through a Social-Collaborative Network (SCN) for decision making and intervention.
Knowledgebase	A knowledge base is an organized, curated collection of information (structured and unstructured) about a particular subject area — a way of making that information more accessible and usable.

Knowledge Graph	The knowledge graph represents a collection of interlinked descriptions of entities – objects, events or concepts. Knowledge graphs put data in context via linking and semantic metadata and this way provide a framework for data integration, unification, analytics and sharing.
Knowledge Representation	Knowledge representation refers to the technical problem of encoding human knowledge and reasoning (Automated Reasoning) into a symbolic language that enables it to be processed by information systems.
Linked Data	The term Linked Data refers to a set of best practices for publishing structured data on the Web. These principles have been coined by Tim Berners-Lee in the design issue note Linked Data. The principles are: Use URIs as names for things; Use HTTP URIs so that people can look up those names; When someone looks up a URI, provide useful information; and Include links to other URIs so that they can discover more things.
Machine Learning	Machine Learning (ML) is a branch of Artificial Intelligence (AI) that aims at enabling machines to learn by themselves from the past experience or data and perform intellectual tasks to solve specific problem that have been traditionally difficult to solved by human beings.
MAPE-K loop	General autonomic computing architecture designed to control the functioning of computer applications and systems without human intervention by using an intelligent control manager. IBM coined this control loop as MAPE-K loop (Monitor, Analysis, Plan and Execute using a shared Knowledge)).
Object-Cebtered Sociality	Object-centred sociality concerned with connecting people via social objects of interest related to their jobs, workplaces and hobbies.
Ontology	The general idea behind ontologies is to make knowledge explicit by expressing concepts and their relationships. In other words ontologies define the common words and concepts used to describe and represent an area of knowledge or collection of information about data and how the data is related. Ontologies are the core concept of Knowledge Representation in

	the Semantic Web. RDFS and OWL are languages that can produce such models.
Resource Description Framework (RDF)	The data modelling language for the Semantic Web. All Semantic Web information is stored and represented in the RDF. RDF provides a flexible mechanism for describing Web resources and their relationships in the form of triples, i.e. subject, predicate and object expressions.
RDF Schema (RDF(S))	Describes the resources with classes, properties and values that could be read and understood by computer applications.
Semantic Formalization	The semantic formalization is often used to interpret complex information which would make information meaningful and accessible to machines. Meaningful and accessible mean that it is possible to make queries based on the purpose of data.
Semantic Reasoning	A semantic reasoner provides the functionality to detect inconsistencies in the ontology model, classification as well as infer new information out of the data.
Semantic Web	The Semantic Web, also known as the Web 3.0, has been described by Tim Berners-Lee as the Web in which computers are capable of analysing all the data with its content, links and transactions between people and computers.
Semantic Web Technology	Semantic Web technologies aim to define and interconnect data in a way similar to that in which traditional web technologies define and interconnect web pages. It enable people to create data stores on the Web, build vocabularies, and write rules for handling data.
Self-managed System	A self-managed software architecture is one in which components automatically configure their interaction in a way that is compatible with an overall architectural specification and achieves the goals of the system.
Self-configuration	Autonomic systems will configure themselves automatically in accordance with high-level policies - representing business-level objectives, for example - that specify what is desired, not how it is to be accomplished.
Self-healing	Autonomic systems will detect, diagnose, and repair localized problems resulting from bugs or failures in software and hardware.

Self-optimization	Autonomic systems will continually seek ways to improve their operation, identifying and seizing opportunities to make themselves more efficient in performance or cost.
Self-protecting	Autonomic systems will protest themselves from malicious attacks and inadvertent cascading failures.
Social Agent	The social intelligence of the agent is the ability to interact with other agents (and possibly with humans) via cooperation, coordination, and negotiation rules.
Social Computing	Social Computing defined as the design and use of Information and Communication Technologies (ICT) that consider the social context.
SIoT	Social IoT (SIoT) is an emerging paradigm of IoT in which heterogeneous IoT devices not only connect and interact together but also socialize and collaborate to achieve a common goal.
SPARQL (SPARQL Protocol and RDF Query Language)	The query language of the Semantic Web. It is specifically designed to query data across various systems. SPARQL is the W3C standard for querying RDF data or linked open data that has capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions.
SWRL	SWRL (Semantic Web Rule Language) a high-level abstract syntax for Horn-like rules that offers efficient reasoning support to ontologies with the benefit of added expressiveness.
Taxonomoy	A taxonomy is a method to classify or categorize a set of terms in a hierarchical structure. In general, it is the study of the general principles of scientific classification. A taxonomy, in general, is semantically weak, because it does not express rich meaning and does not distinguish between aggregation and generalization/specialization relations.
Social Ontology	Social ontology is the study of different categories of social entities and properties including social laws, social facts, social groups, human kinds, social objects and social properties.
Web Ontology Language(OWL)	The schema language, or knowledge representation (KR) language, of the Semantic Web. OWL enables to define concepts so that these concepts can be reused as much and as often as possible. OWL is a knowledge representation

	language designed by the W3C Web Ontology Working Group for use by applications that need to process Web content.
Web 3.0	The Web 3.0 is an extension to the existing Web and its semantics are encoded into web pages which make them transparent in normal use.

CHAPTER 1

Introduction

In the last few years, the convergence of Cyber-Physical System (CPS) and Social Computing has become an active interdisciplinary research topic to implement smart communities. The cyber-physical social system considers human-to-human, human-to-object, and object-to-object interactions within the physical world, human society, as well as in the virtual world. However, several challenges remain concerning appropriate methodologies, frameworks and techniques for implementing socio-cyber-physical relation and interaction for collaborative decision making. The lack of semantic representation of complex socio-cyber-physical relation is one of the main issues for data exchange, accessibility and interoperability between resources. Moreover, it is an ardours task and error-prone for human or traditional software agents to keep track, manage and maintain the larger number of socio-cyber-physical resources and their social dynamics. To resolve these issues, proper design and implementation of a self-managed cyber-physical social system bring the attention of many research communities from the areas of social computing, autonomic computing and Semantic Web technology.

Consequently, the overall aim of this thesis is to develop a semantic social and autonomic framework with a flexible policy model and specification language to support a self-managed and adaptive socio-cyber-physical collaboration and decision making system. The idea is that heterogeneous cyber-physical resources will be represented and semantically connected in a social network and infer information from the retrieved data by applying reasoning rules. The system will also able to monitor and adapt itself according to the changes in the environment and make the collaborative decision with the help of attached intelligent agent called Autonomic Adapter (AA).

This chapter provides a statement of the problem and discusses the main motivation for the research presented in this thesis. This chapter also describes the aims, research questions and challenges with regards to establishing socio-cyber-physical relations, interactions and autonomous decision-making function. It also outlines the significant contribution of this thesis that tries to solve the research problems by developing and implementing the autonomic system within a real-world healthcare system. Finally, this introductory chapter closes with the outlining the structure of this whole thesis.

1.1 Problem Definition

In the last decade, the Cyber-Physical System (CPS) research and its application have shown an enormous societal impact and economic benefit in the many domains such as energy, transportation, manufacturing, health, agriculture and many more. Previous model-based system engineering solution helped overcome various challenges regarding the development of conventional CPS but gaps remain with regards to the development of cyber-physical collaborative system [239]. Cyber-physical collaboration defined as a coordinated process in which objects mutually solve a problem or contribute to a result [92]. CPSs are gradually becoming part of the social-technical society fabric instead of being only just technical systems where human and technical parts are immensely intertwined [65]. With this notion, the convergence of CPS and Social Computing (SC) known as a cyber-physical social system (CPSS) has emerged. CPS defined as a large-scale networked system, which integrates physical and computational components to implement a process in the real-world [262, 208]. On the other hand, social computing concerns the study of human social dynamics in the design and use of Information Technologies (IT) that consider the social context [248]. This emerging technology aims to serve as a base framework for the designing of complex systems that involve a close collaboration between humans and CPS resources [259].

CPSS is a specialised subset of IoT that try to integrate social aspects to the heterogeneous devices where they are socially connected and collaborate to achieve some specific task(s) [2]. In this scope, lots of interesting application domains have been explored ranging from industrial automation to e-health to home automation and to(semi-)automated driving [68]. Particularly, CPS with IoT enabled healthcare applications facilitates remote monitoring of patients health condition in real-time and deliver medication on the fly by sensing and adjusting the actions of medical device [208]. The application of CPS within social context has been exhaustively studied such as smart cities [46], social IoT applications [2], cyber-physical systems convergence with social computing [256], cyber-physical-social convergence in Smart Living [60], cyber-physical social sensing [203], etc. The limitation of these studies has always conducted from a narrative point of view such as focused on the theoretical definition, defined social characteristics, challenges and opportunities, and so forth. These studies lack cyber-physical object-centric social context modelling (i.e., social relations, interactions and communication between objects as well as human), and very limited applications and validations within a real-world context.

The fundamental research challenge is how to represent and manage heterogeneous cyber-physical resources relations and interactions in social-collaborative context. Existing social computing approach provides functionality and features to foster collaboration between human participants only, who work across time, space, cultural and organisational boundaries [148]. A few studies focus on the social object's relation with the human but do not recognise how objects and artifacts themselves can establish and maintain their relationships within a social network [174]. Therefore, it is difficult to create a socio-cyber-physical collaborative system that can support both cyber-physical and social worlds to interact and exchange information to solve problem autonomously.

Object-centred sociality [75] was discussed in a few literatures whereby social structures formed between people as well as shared objects, or their interactions centred around objects [48, 158]. In another study in a Social IoT (SIoT) context indicated that relationships between objects may be established whenever objects collaborate each other to deliver a common services (i.e., emergency response, telemedicine, etc.) [17]. However, these approaches, framework, tools or methodology are not adequate to support socio-cyber-physical collaboration. Therefore, it is necessary to develop a new social framework to overcome the limitation by allowing cyber-physical collaboration and interaction in a social context. Previously, social agent-based solutions [159] introduced for acting as a representative on behalf of the user on social networking tasks, such as to handle digital traces of human activities, to change users' behaviour, produce content and create connections on behalf of them [10]. But, these social agents are not capable to represent and act on behalf of the objects or artifacts for managing and maintaining their social-collaboration task.

Moreover, the lack of semantic representation of socio-cyber-physical object-centric collaborative network limits the data integration, exchange and interoperability between resources. For instance, CPS involves the cooperation of a high number of physical components, explicitly programming the relationships between the system components and considering the number of interrelationships related to failures poses significant challenges [163]. To cope with this situation, more intuitive methods are needed for representing and navigating the heterogeneous information in the network [37]. The Semantic Web ontologies [30] can provide such representation mechanisms by semantically describing these resources, properties and their relationship in a social-collaborative network. Semantic Web technologies play a primordial role by proposing a semantic representation of domain knowledge and by applying a reasoning process to infer hidden knowledge [195]. The notion of ontologies has appeared in Artificial Intelligence that represents knowledge with a collection of definitions of concepts and the share understanding [19]. Consequently, our proposed solution to this problem is a new software agent with an ontological knowledge model that can represent, manage and maintain cyber-physical object's relations and interactions in a social-collaborative network. Ontology-based modelling involves specifying the domain concepts and properties and interrelationships among users, resources and services in the environment [219]. Additional benefits include applying reasoning and inference mechanism that can provide adequate and efficient data integration and deduce new information (i.e., types, relationships, values) from the asserted facts by applying rules [181].

In general, having a large number of resources makes CPS unreliable such as the lack of the abilities to change the system behaviour against unexpected changes in the external environment and to adapt for the internal system failures [105]. Within social context, cyber-physical objects relation may change when they are in different situations, and these changed relations can impact on the state of the cyber-physical object [250]. Thus, manage and track a larger number of cyber-physical resources status and their dynamic social interactions are a complex task for human agents even with the help of traditional social agents. Thus, the newly emerging research problem is how to manage and maintain these large number of cyber-physical resources and their uncertain status within the social world.

Embedding socially intelligent agents into the system can enable sensing social context of resources autonomously and support their collaboration activities by significantly enhancing computation and communication [119]. Social intelligence is the ability to interact and share knowledge with other agents (and possibly humans) via cooperation, coordination, and negotiation [253, 67]. The limitation of existing social agent model is user-centric and not autonomous, it means the owner should control any data stored in the agent, generated through either interacting with other agents, devices or syncing with cloud-based services [10].

To cope with this problem, the combination of autonomic computing paradigm [106] and social networking methodology can provide the solid foundation for building a new socially intelligent autonomous agent with the abilities to manage IT system without human intervention. Integration of social capability with an autonomic computing system to monitor and manage socio-cyber-physical environment autonomously is relatively a new interdisciplinary research area.

In general, autonomic computing borrows concepts and techniques from many fields of Artificial Intelligent (AI) [125, 87] to develop systems that are self-configuring, self-optimising, self-healing and self-protecting [56]. The fundamental idea was to build an intelligent and self-managed system [266] in the same manner as the human nervous system regulates and protects our body [231]. The primary function of an autonomic computing system is to monitor and optimise resources through its sensors, analyses the situations, plan management decisions based on high-level objectives defined in the knowledge and execute the plan through its effectors [56, 98]. Later, the autonomic computing concept has been extended by introducing self-adaptive capability based on context-awareness features that have a positive effect of the system and network operations [215, 164]. The context-awareness idea was to adjust various artifacts or attributes in response to changes and the context of the operating environment at run-time [201].

Since the introduction of autonomic computing, the researcher focuses on addressing self-management capabilities in the areas of software deployment, data storage, resource allocation, communication, and query processing [5]. However, there is a lack of autonomic computing methodology, architecture and framework for designing and developing self-management and adaptive functionality in a socio-cyber-physical system. Despite the several architectural models and practical applications of autonomic computing, there are still many technical challenges which need to be addressed for the successful implementation for self-managed socio-cyber-physical collaboration and autonomous action intervention.

The complexity of managing a large system entails several different autonomic managers that must cooperate to achieve the overall objectives set for the computing environment [168]. The autonomic system needs to be designed with a greater awareness of the fact that the installed autonomic elements can communicate and interact cooperatively with other autonomic elements [125]. Object-centric social networking design can play a vital role to represent autonomic elements or managed resources as social network nodes, where are attached with autonomic adapters can monitor and manage collaborative interaction and communication on behalf of these autonomic elements.

The policy-based approach is arguably the most flexibility and general applicability among the currently popular autonomic system approaches [12]. The common use of fixed and hand-coded policies in a knowledge component is inadequate and inflexible when the system is constantly changing and exhibits varying or uncertain information, which makes the system management decision very difficult and inappropriate [26, 245]. The algorithmic approach based on pre-defined rule sets often fails where the environment is uncertain, unmanageable complexity, emerging behaviour and too complicated to predict [87]. Therefore, policies tend to be developed as prescriptive and externally imposed rules whose enforcement is more flexible for a predictable system behaviour at runtime [36]. There are many policy model have been developed for autonomic computing, including *if-then* rule-based policy engine [199], policy definition language [12], Event-Condition-Precondition-Action-Postcondition (ECPAP) [211], and so on. Event-Condition-Action (ECA) rule [22] based policy enforcement (on an event, if condition, then do action [43]) is a well-known approach for the high-level policy specification of autonomic systems. Existing policy model for autonomic computing is general application-specific and not able to control the behaviour of the system when integrated with social context. Instead, the policy should be specified as externally-imposed rules whose enforcement is for a predictable system behaviour [36] and to control the adaptation behaviour at run-time in a more flexible way, outside the code itself.

Thus, the socio-cyber-physical autonomic system requires well-defined policies that will be configured to specify decision-making criteria to accomplish certain tasks. However, there is no in-depth discussion on fulfilling the gap of appropriate policy specification and execution model for the existing autonomic system methodologies. Though, several researches proposed policy specification languages in general [150, 59, 220] or for the domain-specific autonomic system [199, 204, 21, 117], which are not appropriate for the self-managing socio-cyber-physical system. Therefore, it is necessary to define a flexible policy model that should consider the social-cyber-physical collaborative context regardless of the domain-centric policy model.

1.2 Motivational Domain

The primary motivation of this thesis is to develop an intelligent healthcare system, which can support IoT enabled socio-cyber-physical medical collaboration and autonomous decision making in real-time for supporting vulnerable patients. In general, health information systems are mediated by spatially and functionally distributed resources such as physical devices, software modules, and web resources [54]. For example, in remote healthcare, sensors are used to measure the patient's vital signs and then aggregate these data into the medical database, which are accessed by healthcare professionals [52]. The health monitoring device can detect patients health problem and send alert to emergency responders or family members [195]. Existing system design methodology perform the different healthcare task independently with human support, they are not socially connected, nor capable of autonomous decision making. Therefore, it is a very promising research direction to apply autonomic computing and social computing paradigm together to enable monitoring of socio-cyber-physical medical environment, manage resource collaboration and trigger intervention according to the high-level policy.

Social computing methodology has been successfully implemented in many domains such as scientific, enterprise, healthcare, etc. Particularly, in many areas of healthcare system discussed the benefits of social networking application, such as creating collaborative health awareness using social media [137], remote health monitoring using online social media [127], social media for older adults [142], social networks for future health delivery [95], and many other areas. Healthcare provision is a set of activities that demands collaboration between physicians, nurses, managers, and patients [54] to improve health outcomes, motivate patients, debate healthcare policy and practice issues, to educate and interact with patients, caregivers, and colleagues [243]. In the last few years, the large scale deployment of CPS is becoming research interest in academia, industry and government due to their significant impact on developing socially connected healthcare system.

Additionally, the healthcare information system requires semantic interoperability of health monitored devices to provide accurate and reliable transmission of data across medical networks. The key idea behind using semantic description is to enable representation, formalisation and enhanced interoperability of cyber-physical medical data. The ontology-based modelling of the patient's and physician's context and the available medical devices can provide intelligent alerting of the dedicated physician in case of an emergency through the automatic reasoning of monitored data [108]. The additional benefits may include improved recommendation of health product and services, the discovery of connection with people and resources with similar interests as well as the semantic search of the medical information. Several works such as Lasierra et al. [138] and Paganelli et al. [188] described an ontology-driven solution for a wide range of services such as health status monitoring, real-time alerts and assisting patients at home. Pasquale et al. [63] developed a multiagent system to support the delivery of remote healthcare and Farfan et al. [81] proposed “XOntoRank” system to address the ontology-aware XML keyword search of electronic medical records and other similar work discussed in the literature. Sadly, the agent-based semantic interoperability of the medical connected cyber-physical objects and their data is still poorly represented in existing healthcare system [195]. Therefore, the significant challenge is to connect these medical resources semantically and socially to support cyber-physical healthcare data interoperability, accessibility and autonomous decision making. Our proposed framework in Chapter (4) can address this issue by supporting semantic social connection and collaboration between patient, care communities, emergency departments as well as cyber-physical medical resources and healthcare artifacts.

Recent developments of CPS, IoT medical devices and communication technologies have revolutionised healthcare systems for the diagnosis and treatment of diseases, refereed as “Medical Cyber-Physical System (MCPS)” [140]. For example, CPS has been successfully applied in many healthcare applications for monitoring patient’s conditions continuously and delivers medications on the fly [208]. MCPS is a critical integration of a network of medical sensors and devices that gathers data from the patient’s for cost-effective health decision-making [65]. A general Cyber-Physical Social System (CPSS) is used to collect data from the activities of users on social networks, blogs, or e-commerce sites and then react in some way to those data [261]. However, the limitation of the existing CPSSs [246, 175, 247] are not capable to manage their resources autonomously and cannot adapt to uncertain changes in environment and system failures [105]. Thus, it is a challenging task to implement a healthcare system with self-adaptive goals and to support interactions between the physical and cyber world in assisted-living homes [102]. To improve this situation in the medical healthcare system, our proposed autonomic adapter in Chapter (5) can be embedded into medical cyber-physical medical objects to monitor and retrieve their status, analyse the data and deliver autonomous healthcare services to vulnerable patients or practitioners.

In the healthcare sector, the emerging computational intelligence methods have brought new directions for the development of clinical decision support system (CDSS) to facilitate advanced diagnosis and treatment of diseases [52]. Decision Support Systems (DSS) have emerged on planning emergency response actions in a medical service environment [223]. Decision making is expected to be performed in an implicit and autonomic way rather than the explicit and manual way. Therefore, developing an agent-based intelligent decision support system is important within the medical service community to allow doctors and nurses to gather information faster and processes it to make diagnosis and treatment plans [85]. For example, when a patient is in stress, he or she wants to receive some guidelines before posing a request manually. To fulfil this purpose, there is a great need to increase intelligence in a CDSS, which can monitor and acquire context information and provide relevant services actively to decision-makers or patients. Our autonomic framework with proposed policy model and specification language in Chapter (6) would seem to be most suitable policy configuration approach for CDSS to trigger and execute action by the AA based on the patients' mental health conditions.

1.3 Aims, Research Questions and Contributions

The overall aim of this thesis is to develop a self-managed and adaptive socio-cyber-physical collaborative framework where not only human but also non-human resources can interact, communicate and make collaborative decision autonomously. To achieve this goal, first, we investigate the fundamental theories, computational models, and technological solutions across the semantic social network and autonomic computing paradigm in an interdisciplinary view for socio-cyber-physical collaboration context. We incorporate social computing methodology with ontology knowledge model to semantically represent concepts, properties and relations of human and non-human resources. Second, autonomic computing architectural aspects are studied for re-designing and developing the new socially intelligent agent (Autonomic Adapter) that can be integrated into the socio-cyber-physical collaborative system for achieving self-managed and adaptive functionality. Third, we discuss the existing policy model for the autonomic system, their limitation and the approach to overcome this for managing autonomic adapters behaviour. Finally, we test and evaluate the functionality and feasibility of our proposed approach by implementing a smart healthcare decision support systems involving human and cyber-physical medical resource collaboration with self-adaptive capability.

1.3.1 Research Questions

This study, under the overall aims of this thesis, attempts to address the following research questions:

i) How to establish, manage and maintain heterogeneous socio-cyber-physical resources relation, interactions and collaborations in a social context?

ii) How to semantically represent the relations between human, connected cyber-physical objects and social artifacts and infer new information from the status of the resources?

iii) How to redesign a generic Autonomic Adapter (AA) architecture that can monitor and manage resources and make a collaborative decision with socially aligned values?

iv) What is the best way to define a flexible and consistent policy model and specification language to control AAs behaviour at runtime?

1.3.2 Specific Objectives

Based on the above problem statements, formulated research questions and motivation, this thesis attempts to address the following core objectives:

i) Conducting a literature review of social computing, autonomic computing, semantic web approach, with an emphasis on the methodology, architectural design, modelling and application within the context of the cyber-physical social system.

ii) Develop an agent-centric semantic social-collaborative framework where the software agent will act on behalf of cyber-physical resources to manage and maintain their social interaction, communication and collaboration.

iii) Develop an ontological knowledge model to semantically represent concepts, properties and relations of cyber-physical objects as well as human entities and apply reasoning rule to deduce information from the raw facts by the agents.

iv) Develop a generic autonomic system framework and autonomous adapter architecture to monitor the socio-cyber-physical environment and its resources and perform a wide range of actions according to the high-level policy.

v) Develop a flexible policy model and specification language for autonomic adapters that can be used to define domain-level strategies for collaborative actions and interventions to the environment or managed resources.

1.3.3 Contributions to Knowledge

The works presented in this thesis have made several original contributions to the knowledge by answer the fundamental research questions and associated challenges formulated in this chapter's section (1.3.1). The novel ideas and contributions are briefly listed below and details are discussed in Chapters (4, 5 and 6).

i) **Software Agent-centric Semantic Social-Collaborative Network (Chapter 4)** This chapter proposes an Agent-centric Semantic Social-Collaborative Network (SSCN) that is built on top of the existing social network. The significant advantage over existing systems is that it allows social connections and collaborations not only between humans but also among cyber-physical resources in a social space with the help of software agent. An extended ontology model is introduced that semantically defines the concepts, properties and complex social relations between human and cyber-physical resources in the SSCN.

   The proposed approach is implemented and validated by applying it to a large-scale social-collaborative healthcare service called GRiST [1], used within the United Kingdom. The case study demonstrates the online healthcare aspects to create and manage human and nonhuman resources (i.e., patients, care workers, physical devices, etc.) using the software agent. The domain ontology allows semantic interlinking of these resources and

[1]https://www.egrist.org

the ability to infer when interventions and social collaborations are required to meet healthcare needs. To test the ontology model, the domain ontology is populated with synthesised data from GRiST care domain and perform automated and the Semantic Web Rule Language (SWRL) [2] rule-based reasoning as well as SPARQL queries to test the consistency, correct operation and performance of the ontology. To assess both characteristics and capabilities of the proposed SSCN system, a systematic comparison between SSCN and most recent related work of the area of study is also presented. A Java-based Software Agent API is built that provides functionality to create and manage nonhuman resources and evaluate the effectiveness of the proposed collaborative network. Overall, the ontology testing shows the system is consistent and feasible with regards to emergency notification service based on the data from the cyber-physical medical device and their social context.

However, one of the key limitations of our approach is the inability to handle uncertainty and social dynamics of cyber-physical data due to the complexity and constant changes in resource status as the network grows. Managing and keeping track of these larger numbers of resources is a complex task for human agents even with the help of traditional software agents. Thus, the next research challenge is to design and develop a generic autonomic framework with policy-based autonomic adapter model, which is introduced in the Chapter (5).

ii) **Generic Autonomic Social-Collaborative Framework (GASCF) and Autonomic Adapter (AA) Architecture (Chapter 5)** This chapter proposes a Generic Autonomic Social-Collaborative Framework (GASCF) and Autonomic Adapter (AA) architecture with their behaviour governed by ECA rule-based policy. The GASCF can be used to implement a self-managed IT system such as socio-cyber-physical collaborative system, where numerous AAs can be configured and deployed to manage the environment resources. These AAs can autonomously monitor and analyse the incoming event data and take decisions and actions to the environment based on the policy from the knowledge repository. Our proposed AA architecture is unique compared to other existing intelligent agent model because it is generic and socially intelligent as they can cooperate and communicate with each other through a social network for collaborative decision making when installing in the target IT system. The AAs cannot activate or influence each other directly but should interact and communicate through a social-collaborative network for performing a wide range of actions in a variety of situations. These AAs uses centralised knowledge repository to decide and perform actions on the environment to achieve a shared goal. Within the AA, its *Local Rule Inference Engine* able to trigger actions based on the monitored data and execute some actions through its *Executor*. In some cases, an action trigger by an AA may be insufficient to meet

[2]https://github.com/protegeproject/swrlapi/wiki/SWRLLanguageFAQ

adaptation goal. Hence, the *Global Rule Inference Engine* resides outside but connected with all AAs collate all actions triggered by the individual AA's and then decides the final intervention according to high-level strategy from the knowledge sources.

The cyber-physical health care system is one of the popular application which can greatly benefit from the utilisation of our proposed GASCF and AA architecture. Thus, we demonstrate and validate the proposed system by applying it to the extended GRiST[3] autonomic socio-healthcare service scenarios with deployment of AAs and general implementation of the policies in XML format. Finally, we discuss the uniqueness of our autonomic framework in terms of its new architectural model and the benefits of employing it to manage socio-cyber-physical environment and its resources.

However, the additional work is needed to test and validate the proposed autonomic approach. Because the limitation of this approach is the unavailability and inappropriate policy model and specification language and how the policies would need to be executed by the AAs. This issue directed us to new research direction for developing a flexible policy model and specification language for AA, which is necessary to operate the system without human intervention, presented in the Chapter (6).

iii) **Policy Model and Specification Language for AA (Chapter 6)** This chapter proposes a more comprehensive and flexible policy model and specification language for AA, modelled in XML syntax that can be imposed into autonomic computing knowledge repository. In the literature, several considerable efforts defined policy models and accompanying specification languages based on rules and semantics for the autonomic system [109, 41, 143, 20]. The limitation of these policy-based approaches is fixed coded in general and not re-configurable because they are often very low-level and domain-specific. In this chapter, first, we proposed an ECA rule-based policy execution framework in the socio-cyber-physical environment. Afterwards, we describe an Entity Relationship Diagram (ERD) to illustrate the interrelationship between AA, managed resources and policy components with their associated attributes or properties. Then, we define the Policy Schema Definition (PSD) language and Policy Specification Script (PSS) in XML representation format, a formal way to specify the structure of the policy. The PSD language is unique and the core to implement and validate the structure and consistency of the domain-specific PSS for managing AAs behaviour within the system. The advantage is that it would allow the system administrator to re-configure the policy without modifying the code itself and disrupting the normal system operation. The PSS is an XML instance document that describes the domain-specific policy configuration in terms of PSD syntax and semantics. XML has many advantages to represent the policy for its straightforward extensibility [36] and machine readability. These policies can be deployed as an external script in the knowledge repository and translated as

[3]https://www.egrist.org/home

rules, which can be shared by the AA's intelligent manager components to perform their operation.

We demonstrate and validate the proposed policy model by applying it to the eGRiST CDSS [4] mental-health risk and safety management decision support system [71, 40, 38]. We collected the demo datasets from the eGRiST CDSS in XML format and synthesised based on mental health risk question and assessment and then configured the policies and stored into the knowledgebase accordingly to manage the AAs operation. Finally, we implement a prototype Java console-based application of an autonomic computing engine to test the feasibility of the system. We input the data into the AAs as environmental events and observe the output actions triggered by the AAs according to the rules. The output result shows that the system is feasible, consistent and valid on given input variables that trigger and execute appropriate actions to the environment.

1.3.4 Case Studies

In this thesis, we describe three distinct case studies from a real-word healthcare domain, namely GRiST[5] domain to test and validate the applicability of our proposed system. GRiST (Galatean Risk and Safety Technology) is a web-based CDSS, which was a project sponsored by the European Union within its European Institute of Innovation and Technology Health Knowledge Innovation Community (EIT Health) [6]. GRiST has two independent versions to deliver health services for patients. The initial version is *GRiST older-adult care system* that provides expertise directly to older adults and their network of carer's such as family, friends, and clinical services to ensure they are safe, secure, and thriving [39]. In contrast, *eGRiST CDSS* used for helping practitioners to assess and manage multiple risks associated with mental health problems, including suicide, self-harm, harm to others, self-neglect and vulnerability [260]. It was first introduced by Christopher Buckingham in 2002 [40, 38] and collaboratively developed and maintained by Aston University [7] and Warwick University [8] in the UK. In this thesis, we implement and test our proposed system based on the three different case settings and the datasets from both GRiST and eGRiST systems. These case studies are briefly described below:

- **Case Study 1: GRiST Online Socio-Healthcare Service (Chapter 4).** The first case study explores the potential application of our agent-centric SSCN framework, proposed in Chapter 4. This case study also illustrates the benefits of the ontological representation of healthcare domain knowledge with regards to socio-cyber-physical healthcare resource relation and interoperability in a social space. Currently, within

[4]https://www.egrist.org/home
[5]https://www.egrist.org/home
[6]https://www.egrist.org/grace-age
[7]https://www2.aston.ac.uk/research
[8]https://warwick.ac.uk/research

the GRiST network, human participants interact, communicate and collaborate with each other in different ways to deliver care services to patients. However, the limitation of GRiST and many other similar healthcare systems is that they do not have complete control of social connectivity and activity of cyber-physical objects and they are not semantically described. Thus, to demonstrate and validate the applicability of our SSCN framework, first, we discuss social connectivity of human participant and non-human medical objects (i.e., heart sensor, digital health record, documents, etc.). These resources can be represented as social network nodes and managed by the software agent in the GRiST healthcare network. Second, we discuss the semantic representation of the socio-cyber-physical medical objects and their relation and interactions by constructing an extended "GristOnto" domain ontology in section (4.4.1), which reuses concepts and properties from "SocioCyberOnto" ontology modelled in section (4.2.2.2). Finally, we implement a prototype software agent API with Java programming language that provides the functionally to create and manage GRiST care resources in the network.

- **Case Study 2: GRiST Autonomic Socio-Healthcare System (Chapter 5).** The second case study demonstrates the self-managed and adaptive function of the extended GRiST healthcare system using our proposed approach. This case study discusses real-world healthcare scenario and the intelligent health monitoring and notification functionality of the system by utilising our GASCF and AA architecture, introduced in section (5.2) and (5.3), respectively. The case study illustrates how AAs are attached to the individual cyber-physical resources to continuously monitor their status, analyse the contextual changes, and triggers and executes appropriate actions based on the defined policy. For instance, one or more wearable devices are attached to a patient's body that records their heart-rate and physical activity level and attached AAs monitor their events in real-time. If the threshold parameters coming from the devices are out of range then the AAs trigger some actions based on defined rules. Overall, we discussed the benefits of utilising of our proposed GASCF with AAs, which can minimise the GRiST social care management complexity, reduce human intervention and provide intelligent notification for delivering instant care and services as appropriate.

- **Case Study 3: eGRiST Clinical Decision Support System (CDSS) (Chapter 6).** The third case study illustrates the practical application and configuration of our proposed policy model and specification language to manage AAs behaviour when integrated into eGRiST CDSS. The eGRiST CDSS is designed based on a cognitive model of human expertise manifested by a sophisticated hierarchical knowledge structure or tree for mental illness diagnosis. In eGRiST CDSS, risk classes are represented by hierarchical knowledge structures or trees called galateas, which are used to represent mental-health expertise. The root node of the tree is the risk node, which is deconstructed into sub-nodes that are themselves trees until the leaf nodes are reached, representing the input data. The data used for input to the tree can be any type, which is then converted into a fuzzy-set membership grade, called Membership Grade (MG) and the MG score is between 0 to 1 [40].

Currently, the system handles various action types using a static database design with one-table-fits-all solutions, which is not convenient in many cases. The drawback is that for every new action type that is added to the system, new tables, particular to that action need to be added, which is very inefficient. Thus, it is crucial to improve the reliability and performance of this system in terms of actions management by integrating AAs into every assessment node (i.e., cues). Thus, this case study is appropriate to illustrate AAs integration and autonomous actions and interventions using our policy model for autonomous mental health risk assessment in eGRiST system. The events triggered by the MG of a specific node can be handled and processed by AAs for action intervention according to the policy specification. The overall benefit of using our AA and policy model is to implement actions or interventions plan that will be executed based on the mental health risk assessment answer submitted by patients or practitioners in the eGRiST system.

1.4 Thesis Structure

The work produced in this thesis has been conducted in a sequential manner whereby solving one problem led to the identification of another pressing research challenge. This consecutive research workflow is translated to logical connections of the thesis components, summarised in Figure 1.1, which shows how each part relates to another part. The peer-reviewed published papers [114, 113, 112, 111] related to the thesis are listed in subsection (1.5). The structure of the thesis is as follows:

- **Chapter 2** presents a comprehensive background study of the social computing approach, opportunity and limitation for supporting cyber-physical object interactions and collaboration. This chapter also reviews the Semantic Web technology and the advantage of ontological knowledge model to semantically represent the cyber-physical concept, properties and relation in a social context. This chapter also outlines the background of the autonomic computing paradigms, different application areas and current challenges in the existing architectural design. Finally, this chapter reviews the policy-based management of the autonomic system and explains the issues with regards to the existing policy model and specification languages for supporting self-managed socio-cyber-physical collaboration.
- **Chapter 3** discusses the state-of-the-art related work within the areas of the semantic social network, cyber-physical social system and policy-based autonomic computing.

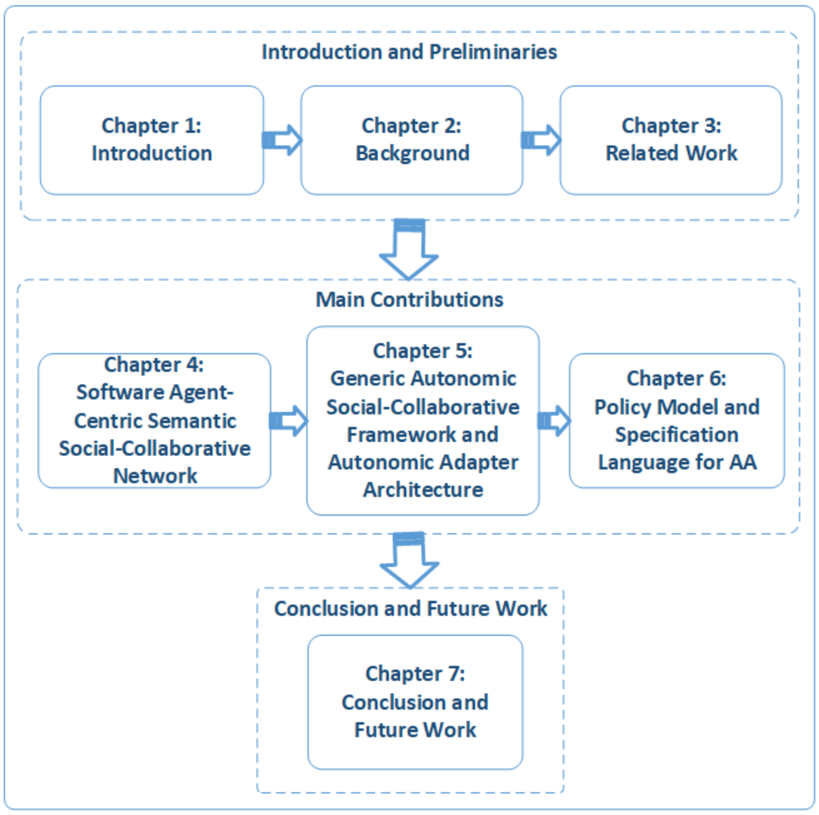

Introduction and Preliminaries
Chapter 1: Introduction
Chapter 2: Background
Chapter 3: Related Work
Main Contributions
Chapter 4: Software Agent-Centric Semantic Social-Collaborative Network
Chapter 5: Generic Autonomic Social-Collaborative Framework and Autonomic Adapter Architecture
Chapter 6: Policy Model and Specification Language for AA
Conclusion and Future Work
Chapter 7: Conclusion and Future Work

Fig. 1.1 Thesis components and research workflow

This chapter particularly reviews the existing research work and their limitations with regards to the developed methodologies, architectures, techniques and their applications in many real-world aspects.

- **Chapter 4** introduces a Software Agent-centric Semantic Social-Collaborative Network (SSCN) model with an extended ontology model. This chapter discusses the implementation of a domain ontology and Agent API to represent and manage cyber-physical resources relation and interaction in a social network environment. Finally, this chapter illustrates the evaluation of the system by ontology testing in terms of syntax, semantics, and rule-based reasoning within real-world GRiST healthcare scenarios.

- **Chapter 5** presents a new Generic Autonomic Social-Collaborative Framework (GASCF) and Autonomic Adapter (AA) architecture. This chapter also discusses self-managed and adaptive functionality of GRiST healthcare system through the realistic implementation of the proposed GASCF and AA architecture with policy configuration.

- **Chapter 6** proposes a flexible policy model and specification language for AA. This chapter discusses the policy execution model, PSD language and PSS that provides the syntax and structure of the policy definition and XML representation of the domain policy adheres to PSD. The real-world eGRiST CDSS case study is discussed to demonstrate the practical implementation and configuration of the policy model for mental health risk assessment and trigger appropriate actions by the AA. The system is tested and evaluated by implementing Java-based AA engine and integrating it to eGRiST CDSS and observes inputs and output actions by the system.

- **Chapter 7** concludes the entire research work pursued. The main contributions of the thesis are pointed out and directions for future research also discussed in this chapter.

1.5 Publications

The parts of the research work that has been done towards this thesis have been presented and published in international peer-reviewed journals, conferences and workshops as listed below:

- Hussain, N., Wang, H. H., Buckingham, C. D., & Zhang, X. (2020). Software Agent-Centric Semantic Social Network for Cyber-Physical Interaction and Collaboration. *International Journal of Software Engineering and Knowledge Engineering.* 30(06), 859-893.

- Hussain, N., Wang, H., & Buckingham, C. (2019). Artifact-Centric Semantic Social-Collaboration Network in an Online Healthcare Context. *In 17th International Conference on e-Society.* IADIS, Utrecht, Netherlands.

- Hussain, N., Wang, H. H., & Buckingham, C. (2018, April). Policy based generic autonomic adapter for a context-aware social-collaborative system. In 3rd International Conference on Intelligent Systems and Computer Vision (ISCV) (pp. 1-9). IEEE Digital Explore.

- Hussain, N. and Wang, H. (2014). Semantic Enabled Social-Collaborative Research Framework for Proteomics Domain. *In Third ASE International Conference on Social Informatics.* Harvard University, USA, December 14-16.

CHAPTER 2

Background

This chapter presents a comprehensive background study of the research presented in this thesis. This chapter reviews the most relevant theories, model, architecture, application and limitation of cyber-social computing, autonomic computing and Semantic Web technology in a multidisciplinary view. We precisely discuss the research areas include the social network for cyber-physical communication and collaboration, Semantic Web technologies to semantically represent socio-cyber-physical relation and interactions, and policy-based autonomic computing for developing self-managed and adaptive socio-cyber-physical collaborative system.

2.1 Socio-Cyber-Physical Collaboration

2.1.1 Social Computing and Networking

Social Computing (SC) defined as the design and use of Information and Communication Technologies (ICT) that consider the social context [248]. In the literature, social computing or Web 2.0 was characterised as collaborative projects, blogs or microblogs, wikis, content communities, social networking sites, and virtual worlds that are serving functions for different purposes [57, 243]. Social networking system uses Web 2.0 technology [118, 144], which allow quick, easy and wide communication between users for personal or professional reasons [161, 90, 65].

A social network graph composed of individuals, organisations or social objects, called nodes, which are interrelated or connected through common social relationships (i.e., friendship, following, likes) either directly or indirectly called edges [161, 90]. According to Arnaboldi et al. [15] and Tang et al. [234], a typical way of representing a social network as a graph $G = (V, E)$, where V is the set of vertices representing the users in the social network, and E are the set of edges connecting users representing the social relationship. For example, users can be represented as graphs with nodes or vertices, and their social links to friends or others can be represented as edges, and leverage their social links to share content, organize events, and search for specific users or shared resources, communication, collaboration, or even for virtual friendship [252, 145].

2.1.2 Object-Centred Sociality

At the beginning of social network study, interactions between humans have long been considered an important part of social context. Later, a new dimension of social network introduced as object-centred sociality in a few literatures, whereby social structure formed between people and shared objects, or their interactions centred around objects [48, 158]. Object-centred sociality first introduced by Jyri Engeström [75] to examine how the inclusion of shared objects can enhance online social networking between people and shared objects. It describes the fact that strong social relationships are built mainly when individuals are connected together through the objects they create and collaborate around a shared object in online social networks [120, 174]. Individuals or groups can connect and engage with objects, and receive a data stream of events that are deemed of collaborative importance by the business and create focused conversations related to the object [126]. For instance, a published paper is an object that connects the researchers that authored it electronically using a word processor program, and if other people read the paper or download it, the author will be notified. The research suggested that people may not only to connect each other, but they also connect through shared objects or artifacts (i.e., documents, tasks, etc.), which can be represented as graphs with nodes both for people, shared resources and work artifacts [24]. For example, an early study of software engineering coordination showed that developers can be connected to one another through their work artifacts [47].

Artifacts in a wide sense broad class of instruments, objects and tools that are used by actors in the distributed task environment to configure and facilitate group decision-making, thinking and communication [226]. When the artifacts, in general, are viewed from the perspective of their use, they can both support communicative and instrumental activities, and they can mediate our activity towards other humans or towards ‘objects’ [154]. Artifacts are classified as physical artifacts that serve an important role in the sequencing, triggering, closure of a task, whereas cognitive artifacts assist in representing task knowledge, procedures and help transcend barriers to problem-solving and decision-making [226]. In contrast, social artifacts defined as any form of online content (i.e., post, connect, recommend, tweet, tag, and endorse) that are created in a Web 2.0 application [156].

2.1.3 General Social-Collaborative Aspects

Collaboration is defined as the process of working together between people, groups and organisations in a sociable environment to systematically solve a problem that could not be solved by an individual alone [28]. A collaborative network defined as a collection of businesses, individuals and other organizational entities that possess the capabilities, and resources needed to achieve a specific outcome [214]. Collaboration environments typically provide a range of benefits include the basic requirements of communication, coordination, collaboration and various mechanisms for information provision and access [100]. In the beginning, there was drawing attention on the expertise, social and computer scientists for computer-supported cooperative work [94] and looked at how groups worked, and seek to discover how technology can help them work together [73]. Thus, a new adequate Collaborative Work System (CWS) had emerged [31] to facilitate individuals and organisations to manage, communicate and work together across time and space barriers [83]. Groupware is a class of Computer-Supported Cooperative Work (CSCW) that enables individuals to collaborate on projects with a common goal from geographically dispersed locations through shared Internet access as a means to communicate within the group [45].

Moreover, collaboration support is not only about providing technologies and tools but also about shaping socio-technical systems [130]. Early research suggested that the technical system and the social system have to be co-optimised for the whole system to be successful [82]. The technical system helps the people to communicate and collaborate with each other, whereas the social system facilitate to building relationships between the group members and their formal and informal social actions and interactions. The socio-technical network gives a structure to represent the flow of information among actors through the exchange of work artifacts [145]. The development of Web 2.0 technologies revolutionised the potential to harness online collaboration more interactively by increasing easy communication, social interactions and active engagement as well as reducing the cost, time and space barriers. Web 2.0 also known as social software [210] refers to a collection of electronic, web-based, community-centred applications and technologies [151] that facilitate interactive information sharing, user-centred design and collaboration [225].

With the advent of Web 2.0, social computing has emerged to understand the collaboration and interaction of human observed via socio-technical systems such as online social networks and web-enabled applications. Social computing refers to the computational facilitation of social studies and human social dynamics, as well as the the design and use of Information and Communication Technologies (ICT) that consider the social context [248]. In the literature, social computing tools characterised as collaborative projects, blogs or microblogs, wikis, content communities, social networking sites, and virtual worlds that are serving functions for different purposes [57, 243]. For example, most popular open social media applications including Facebook, Twitter, LinkedIn, etc. built on Web 2.0 technologies, which enable and encourage participation, collaboration, conversation, openness, creation and socialisation among a community of users [88]. Social media users connect with others by sharing user-generated content and information, following their updates, responding and commenting on them in real time [116, 243, 186].

2.1.4 Cyber-Physical Social System

Cyber-physical system (CPS) is a large-scale and networked system, which consists of physical and computational elements and currently of interest in academia, industry and government due to their potentially significant impact on society, environment, and economy [262]. CPS are typically concerned with the sensing and control of physical phenomena through networks of interconnected devices to achieve defined goals [62]. In recent years, CPS paradigm shifted to Cyber-Physical Social Systems (CPSS) [46] that have become an active interdisciplinary research topic to deliver services that are autonomous, intelligent, connected and collaborative. Within the context of CPSS, the tight integration involve human-to-human, human-to-object, and object-to-object interactions in the physical world, social world, as well as in the virtual world [203]. In a collaborative cyber-physical systems resources can interact with each other and form groups to achieve joint goals [239]. Using a social network, humans and cyber-physical resources can jointly perform task that the resources itself are not able to do it separately [224]. In this scope, lots of interesting application domains have been explored ranging from industry automation to e-health to home automation and to(semi-) automated driving [68], and indeed as part of smart infrastructures and cities [236] to improve the quality of life of the smart city citizens by providing intelligent and personalised services in a wide variety of aspects.

Cyber-physical collaboration is considered as the third generation of online social networks that consider social relationship, interaction or connection among physical objects, includes human being, smart items and general physical objects using the sociological point of view [250]. As most CPS involve the cooperation of a high number of components, explicitly programming the relationships between the system components and considering the quantity of interrelationships related to failures poses significant challenges [163]. However, there is a lack of discussion on how the social relationships should be established by cyber-physical objects and does not propose any solution regarding the required architecture and protocols. Due to the enormous amounts of devices interact with each other socially, currently used approaches or architectures is not be sufficient to deal with the arising complexity such as social relation, collaboration and autonomy. To fulfill their tasks, the CPS components rely on semantic representation of their environment to enable the CPS components to agree on the meaning of common concepts they use in an open environment, e.g., an ontology describing the concepts and relations within the application domain [97, 93].

2.1.5 Agent-Centric Social Environment

Agents and artifacts meta-model are introduced in the context of agent-oriented software engineering for modelling complex software systems [58]. The term ‘agent’, ‘software agent’ or ‘intelligent agent’ are often used interchangeably in the literature and defined as an entity that reacts to changes in its environment through a reasoning process [155]. Social agents have been introduced in research to achieve automated or semi-automated social interactions pervasive social networking. An agent can be a software (e.g., daemon security agents), a hardware component (e.g., thermostat), or a combination of both (e.g., a robot), whereas the IT system (i.e., traffic monitoring network) refers to the place where the agents are attached [67]. Typically, agents are placed in an IT system that senses different event parameters to make a decision based on the goal of the managed entity [218].

A social agent is described as an artificial agent with capabilities to change users' behaviour, produce content and create connections over existing social networks, and collaboration with other agents [10]. A user's surrogate agent provides a personalised interface between the user and the environment, being knowledgeable both about the environment and of the user's interests and preferences, and enhances operational capabilities in the environment [207]. Agents often collaborate towards a specific goal and need to communicate and share results with other agent [155]. Acting as a representative of the user on social networking tasks requires prior knowledge of the environment and understanding of user's behaviour, but also a set of social rules and goals according to which an agent would initiate social interactions [10].

In agents and artifacts meta-model, agents are the (pro-)active entities in charge of the goals/tasks that together build up the whole multi-agent system (MAS) behaviour, whereas artifacts are the reactive entities providing the services and functions that make individual agents work together in a MAS [184]. The artifacts are used by the agents in the distributed task environment to configure and facilitate group decision-making, thinking and communication [226]. Ontologies allow the sharing of common knowledge among people and software agents that represents collaborative tasks and enable queries and inferences about shared tasks [205]. Agents and artifacts meta-model relevant to CPSs are key steps to support collaboration in social context.

2.2 Semantic Social Network

Semantic Web Technology (SWT) is proven to be used for explicitly model online social interaction between participants [77]. Its automated aggregation capability of user's distributed social connections give a better illustration of user's profile and improve the functioning of the web applications [157]. Developers uses SWT to augment to create, reuse, and semantically link content on social networking and media sites [37]. For example, ontologies enables community web portals more efficient at the task of sharing information and offers high quality searching features by providing semantic-based browsing [177]. Ontological knowledge graph widely applied in content publishing, navigation (link content and allow exploration of topics) via schema.org [1] in many areas such as Yahoo!, Bing, Google, BBC, and so on.

[1]https://schema.org/

2.2.1 Semantic Knowledge Representation

Knowledge representation and knowledge engineering play an important a role in Artificial Intelligence (AI) research. The traditional methods lack of considering the semantic relations between words, which is difficult to improve the accuracy of these classification methods. Many of the problems that are expected to solve by machine require extensive knowledge about the world.

The Semantic Web (SW) is an extension of the current web, also known as the Web 3.0, the Linked Data Web, the Web of Data, has been described by Tim Berners-Lee [30]. The main goal of the SW is to improve the processes of knowledge representation (KR) and information retrieval (IR) on the web [7]. SW introduced to connect information as a web of linked data, intended to be accessed and understood by computers, rather than by people [123] so that they can perform increasing sophisticated tasks. Current web is unstructured, connected by hyperlinks (a web of linked documents). Whereas SW enables web data to be linked from one source to another (i.e., people, places, time periods, concepts, etc.) in a structured form [232] and well defined meaning [166]. The World Wide Web Consortium (W3C) [2] is the main international standards organisation for the SW and develops recommendations for representing, sharing and reusing knowledge. We discuss the main SW standards below:

2.2.1.1 XML and XML Schema

Extensible Markup Language (XML) is a simple, very flexible text-based format derived from Standard Generalized Markup Language (SGML; ISO 8879:1986) [3] that provides mechanisms for describing document structures using markup tags. XML originally designed to meet the challenges of large-scale electronic publishing. The current development plays an increasingly important role for the exchange of a wide variety of data on the web. However, in terms of semantic interoperability, XML has restrictions. For instance, there is no way to recognise the semantics of a particular domain because XML aims at a document structure and enforces no common interpretation of the data [44].

The XML Schema [4] is introduced to formally specifies the structure of an XML document that gives additional semantic meaning to the data, assigned to different elements and attributes and use to assess the validity of well-formed XML document. The Document Type Definition (DTD) [5] or XML Schema Definition (XSD) [6] are the two main schema definition languages for XML that can are commonly used to define the syntax and structure of an XML document.

[2]https://www.w3.org/standards/semanticweb/
[3]https://www.w3.org/XML/
[4]https://www.w3.org/XML/Schema
[5]https://www.w3schools.com/xml/xml_dtd.asp
[6]https://www.w3.org/TR/xmlschema11-1/

2.2.1.2 RDF and RDF Schema

Resource Description Framework (RDF) [7] is a standard data modelling language for the SW. It has been developed as solution to represent information about resources on the WWW (e.g., meta data/annotations), data integration and interoperability purpose in the WWW [44]. It describe web resources and their relationships in the form of triples (i.e., subject, predicate, object) expressions [241], so it could be read and understood by computer applications.

The subject is the resource or the thing (i.e., Eric) that is being described, and predicates express the relationship (i.e, type) and object (i.e., Person) is the value assign to the predicate [44]. A resource is uniquely identifiable by a Universal Resource Identifier (URI) and central to the SW [29] and the property value is either also a URI or a literal (e.g., String, Boolean, Float). A triple can graphically be described as a directed arc, labelled by the property (predicate) and pointing from the subject node to the property value node [136]. However, the RDF data model provides very few mechanisms to address the problem of understanding the meaning of the terms used in semantic annotation. Hence, RDF Schema (RDF(S)) has been designed as an extension of RDF to overcome the limitation of RDF specifications.

RDF(S) [8] describes the resources with classes, properties, and values [44]. Although, RDF(S) allows the hierarchical specification of classes and properties that enable simple inference, the expressive power of RDF(S) is quite limited and unable to support many commonly required features, such as negation or disjunction.

2.2.1.3 DL and Ontology

Description logics (DL) is a knowledge representation language to represent the knowledge of an application domain in a structured and formally well-understood way. DLs differ from their predecessors, such as semantic networks and frames, in that they are expressive and equipped with a formal, logic-based semantics [19].

An ontology is one of the main building blocks of Semantic Web that build on RDF/RDF(S) and add expressiveness to the ontology. An ontology is a collection of definitions of concepts and the shared understanding comes from the fact that all the agents interpret the concepts [19]. It formally and explicitly specifies entities, attributes and properties related to a domain [180]. According to W3C [9] standard definition, in an ontology model, *Classes or Concepts* represents a collection of objects or individuals (usually sharing some common characteristics), *Properties* describes the relationships between individuals (and data) and *Instances* are concrete members of a class. There are two types of properties defined: object property that relates individuals to other individuals, and data property that relates object or individuals to data values.

[7]https://www.w3.org/RDF/
[8]https://www.w3.org/TR/rdf-schema/
[9]https://www.w3.org/standards/semanticweb/ontology

There are many languages have been developed to build an ontology for different purposes. The Ontology Web Language (OWL) [10] is the schema language or knowledge representation (KR) languages, of the SW. OWL enables authoring ontologies to represent rich and complex knowledge about things or groups of things [241]. The OWL standard defines different syntaxes based on RDF(S), XML and proprietary text format, what has greater machine interoperability of the web content. Similar to RDF(S), OWL provides syntactic modelling constructs for the basic elements of an ontology, i.e. concepts, relations and instances. In OWL, these are called classes, properties and individuals, respectively, and they correspond to concepts, roles, and individuals in Description Logics (DL) [96]. OWL abstract syntax serves as a human readable text format to present OWL ontologies to knowledge engineers.

2.2.2 Rules and Reasoning

Rules are a key element of the Semantic Web that provide a foundation for reasoning capabilities for underpin the intelligent manipulation and exploitation of information content. Reasoning is the process to deduce new information from a set of asserted statements using rules. An inference engine deduce logical consequences from a set of facts or detect inconsistencies in the ontologies [185] by using the technologies such as description logic theory, forward chaining, backward chaining, and Rete Algorithm, and so on [53]. The most popular automated reasoning engines like Pellet [11], Racer [12], FaCT++ [13] are based on very expressive OWL DLs. The OWL language itself is designed to support various types of inferences such as subsumption and classification [210].

The reasoning tasks can also be done through the application of the rule languages such as RDF Query Language (RDQL), SWRL (Semantic Web Rule Language) [205], or SPARQL (Protocol And RDF Query Language) [232]. The SWRL [14] is designed based on a combination of the OWL DL and OWL Lite sublanguages of the OWL with the Unary/Binary datalog RuleML sublanguages. SWRL includes a high-level abstract syntax for Horn-like rules that offers an efficient reasoning support to ontologies with of the benefit of added expressiveness. SPARQL [15] is the W3C standard for querying RDF data or linked open data that has capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions.

There are few free and open source Java framework available for building Semantic Web and Linked Data applications. For example, Apache Jena [16] ontology inference system is designed to support a wide range of inference task or reasoners. Jena includes a general

[10]https://www.w3.org/OWL/
[11]http://pellet.owldl.com/
[12]http://www.ifis.uni-luebeck.de/ moeller/racer/
[13]http://owl.cs.manchester.ac.uk/tools/fact/
[14]https://www.w3.org/Submission/SWRL/
[15]https://www.w3.org/TR/rdf-sparql-query/
[16]https://jena.apache.org/

purpose rule-based reasoner (Generic Rule Reasoner) which is used to implement both the RDFS and OWL reasoners but is also available for general use [192]. RDF Data Query Language (RDQL), a declarative query language for RDF supported by Jena's inference models. RDQL uses a declarative SQL-like syntax for querying information contained in an inference model, often expressed as a set of triples. On the other hand, the OWL API [17], a high-level Java Application Programming Interface (API) for working with OWL ontologies that has been around for more than a decade. It supports parsing and rendering in the syntaxes defined in the W3C specification (Functional Syntax, RDF/XML, OWL/XML and the Manchester OWL Syntax); manipulation of ontological structures; and the use of reasoning engines [107].

2.2.3 Existing Ontologies

Numerous ontology-based reasoning framework has been developed in the community [6], [224], [239] for addressing cyber-physical system or social networking application in many domains. However, existing cyber-physical or other social collaboration ontologies are not applicable in practical, not reusable or inadequate to semantically represent cyber-physical object relations and interactions in a social context. Most of the existing social ontology modelling focuses on the study of different categories of social entities and properties, including social laws, social facts, social groups, human kinds, social objects and social properties according to social theorists [76]. Research explored that social software would benefit from utilising existing social ontologies by modelling and visualisation of social relation and interaction for the web. For example, FOAF (Friend-Of-A-friend), SIOC (Semantically-Interlinked Online Communities), SKOS (Semantically-Interlinked Online Communities), and DC (Dublin Core) ontology are the well known social ontologies that can provide a better-personalised recommendation based on semantic user profile [72], semantic search and data reuse [160], social network analysis [181], modelling social contexts for pervasive computing environments [33], and so forth.

FOAF ontology [18] is the most popular social ontology used for describing people, their relationships and social activity [146, 77]. FOAF has become a widely accepted standard vocabulary for representing social networks, and many large social networking websites use it to produce semantic profiles for their users [91]. SIOC ontology [19] provide a semantic layer to the social web platforms to describe the most common resources exchanged within and across the web of communities such as define the concept of “site”, “post”, “forum”, “blog”, etc. [146, 77, 64]. SIOC also allows adding metadata to blog posts, as it is strongly mapped to existing ontologies as Dublin Core or FOAF for the people aspect [191]. SKOS ontology [20]

[17]https://www.w3.org/2001/sw/wiki/OWLAPI
[18]http://www.foaf-project.org/
[19]http://sioc-project.org
[20]https://www.w3.org/2004/02/skos/

describes systems of organisation of knowledge [146]. A common practice is to use DC terms and its refinements to add librarian metadata to the post such as creation date using the term dc:created, title (dc:title) or author (dc:author) [191]. The Figure 2.1 by Ereteo, et al. [77] shows the alignment of most common social ontologies that is reused in many social application.

Fig. 2.1 Alignments between SIOC, FOAF and SKOS [77]

Recently, the W3C Semantic Sensor Network (SSN) ontology [21] introduced that can describe sensors and actuators in terms of capabilities, measurement processes, observations and deployments [55]. However, a little explanation is given in the literature addressing the problem of agent-centric cyber-physical collaboration in social environment.

2.3 Autonomic Computing Concept

Autonomic Computing (AC) is a generic concept that has becoming an important software engineering discipline with rich literature, active research communities, and a growing number of applications [87]. AC introduced by Paul Horn at IBM, in 2001 [106] to develop systems endowed with autonomic behaviour, capable of self-government and self-organisation without human intervention [152]. AC field itself draws from a disparate number of well established fields, including artificial intelligence, telecommunications research, mathematics, software/hardware engineering, statistics etc., each with its own view of how a system is to be defined [213]. Particularly, Artificial Intelligent (AI) is an appropriate core discipline from

[21] https://www.w3.org/TR/vocab-ssn-ext/

which autonomic computing borrow concepts and techniques, as automated decision making is its central focus [125].

The grand challenge of autonomic computing is to reduce the increasing scale and complexity of IT infrastructure by making systems and applications self-managing [227]. Complex systems are defined as those with much diversity in the interacting components and little certainty about the outcomes in advance [179]. The complexity normally arise due to the several factors, including advances in hardware and software technology, the growth of Internet-connected networks, etc. AC goal is to handles such system complexity and uncertainties by realising that computing systems and applications capable of managing themselves with minimum human intervention [5]. For several years, there has been a fair amount of research working on conceptual architectures or theoretical designs to create software systems that can change themselves in the context of system performance, security, and fault management issues. However, there is still a lack of understanding between research communities of the broader software engineering aspects of autonomic system development.

2.3.1 AI and Machine Learning

AI concerns the study and design of intelligent machines or software that are capable of performing a similar task of using computers to understand human intelligence [162]. AI has been gaining popularity due to the three major factors: the growth of Big Data; the availability of cheap, scalable, computational power; and the development of new machine learning techniques [187]. Machine Learning (ML) is a branch of Artificial Intelligence (AI) that aims at enabling machines to learn by themselves from the past experience or data and perform intellectual tasks to solve specific problem that have been traditionally difficult to solved by human beings [172]. The current research in machine learning focuses on computer vision, hearing, natural languages processing, image processing and pattern recognition, cognitive computing, knowledge representation, and so on [167]. Machine-learning techniques can help in automating the time-consuming process of knowledge acquisition that is essential to the development of a knowledge-based system [194]. More specifically, one popular view of the field of AI—outlined by Russell and Norvig 2003 [200] in their standard text on AI that it is fundamentally concerned with the design of rational agents. Knowledge is often acquired by machine learning techniques, which are usually uncertain, for example in a probabilistic sense or in the sense of fuzzy logic [96].

2.3.2 Self-Managed and Adaptive System

A self-managed system considered to automatically make decisions and take actions at runtime that relieving the system administrator from detailed and manual operation activities [267]. Self-management concept extended to the self-adaptation aiming to adjust various artifacts or attributes of a system with the ability to adapt, manage, repair and update themselves autonomously in response to changes in run-time and within the context of the operating environment [21, 201]. In contrast, self-adaptation is the ability of a system to adapt, manage, repair and update themselves automatically at the run-time often achieved through a feedback control loop [21]. In the IBM autonomic computing architecture, self-managed resources manage their behavior in response to higher-level goals and interact with other resources to provide or consume computational services [56]. At the adaptive level, not only does the system monitor, correlate, and develops action plans, the system also takes corrective actions according to established policies [4]. However, enabling adaptive and flexible functionality when the system is constantly changing and exhibits varying or uncertain information, performance may be poor [66].

2.3.3 Autonomic Computing Properties

AC theory encompasses four common self-management properties usually categorised as self-* properties or simply summarised under the term self-managing by several studies [56, 229, 125, 89, 4]. The key requirement for the self-* capabilities is that the system is aware of its own state, requirements, or context at runtime and described as follows:

- self-configuring is the ability to adapt and re-configure automatically (i.e., installing, updating, and integrating software entities) in response to the changing environments by using high-level goals. The changing environment could be an integration of new components or the removal of existing ones.

- self-healing is the ability to detect errors at runtime and initiate policy-based corrective action without disruption of system services (i.e., detect, diagnose, and repair localised problems resulting from bugs or failures in software and hardware). IT vendors devote a lot of their time to identify, tracing, and determining the root cause of failures in complex computing systems. Self-healing mechanism detects, diagnose and initiate remedial action by switching to a redundant component or by downloading and installing software updates.

- self-optimising is the ability to monitor and tune resources automatically (i.e., reallocating resources, dynamic workload management, interfacing with other automatic modules to exchange data and files). This environment continuously learns the ways to improve its operation, identify the opportunities to make the system efficient in performance. For example, the environment will proactively seek to upgrade its function by finding, verifying, and applying the latest updates.

- self-protecting system has the ability to anticipate, detect, identify, and protect themselves from attacks from anywhere (i.e., resolve open security concerns, automatically backup and recover resources) and take autonomous corrective actions to make itself less vulnerable to attacks on business data and general failures.

Fig. 2.2 IBM Autonomic Manager

Additionally, autonomic computing applications required some other properties apart from the generic self-X attributes, which are interrelated and contribute to an appropriate achievement of the AC goals [129]. These properties may include adaptability, awareness, monitoring, dynamicity, autonomy, robustness, mobility, traceability. According to Sterritt et al. [229] and Duan [69], other attributes may include self-awareness (aware of managed resource) - a system must be aware of its internal state, ultimate capacity, physical connections with other systems to govern itself [99, 242], and context-awareness (aware of the external environment) - the system knows how to negotiate, communicate and interact with the systems or other components of a system, and how to anticipate environmental system states, situations and changes [242].

2.3.4 Intelligent MAPE-K Loop

A classic IBM autonomic computing architecture is generally composed of an *autonomic manager* that controls one or more managed resource or element (i.e., any software or hardware resources) using a *sensor interface*, which collects information from managed resource and *effectors* that carry out changes interface [266, 245]. IBM coined this control loop as MAPE-K loop (Monitor (M), Analysis (A), Plan (P) and Execute (E) using a shared Knowledge (K)) [56] and their interactions together produce a self-managing system as shown in Figure 2.2 that control the functioning of computer applications.

In general, *Monitor* component collects, aggregates and filter information retrieve from the managed resources or environment through sensors (i.e., metrics and topologies), *Analysis* component correlate and model complex situations (i.e., time-series forecasting and queuing models), *Plan* component constructs the actions needed to perform a desire action using policy, and *Execute* component control the execution of a plan with considerations of dynamic updates and apply changes through effectors on the controlled system [56]. The *Knowledge* is the core component in an autonomic manager that deposit the sharing knowledge of data, including system logs, strategies and some measures used by the MAPE components [69]. The knowledge is related to any information that enables the provision of self-adaptation, such as historical information about the use of the system, context information, managed resource details, metrics, symptoms and predefined policies [21].

At the most abstract level, the resource managed and control in the MAPE-K loop is the smallest functional unit that exists in the run-time IT environment. These managed elements or resources may include individual computers, servers, and data-storage devices, but they also may include operating systems, commercial software applications, and the middleware that connects and integrates them [170]. The key idea of this control loop is to foster self-* properties within the software systems [80] to design safe, efficient and predictable controllers, which enable monitoring and adjusting system's operation dynamically while keeping overhead low [266]. Fundamentally, this loop acts upon some hand-coded logic, embedded by management experts in knowledge, which is not appropriate in many context [26].

2.3.5 Social Intelligence of Autonomic Agent

Autonomous software agent focuses on the interactions between the decision processes within the software entity and its surrounding environment to react to changes in their execution contexts [80]. It is an autonomous component that encapsulates the logic and control of its execution, which is directed towards the achievement of some task(s), as described in the agent program [196]. The attributes of an agent are autonomy, sociability, reactivity, pro-activeness, adaptiveness, interactivity, rationality, and interactivity, etc [254]. The social intelligence of an agent has implied interaction with other agents via cooperation, coordination, and negotiation plus intelligence [70, 253]. A social agent is described as an artificial agent with two dimensions: processing capabilities and differentiated knowledge of the self, task domain, and environment that drive to the development of a social behaviour [10]. However, traditional agent model cannot activate or suppress each other directly, they must use accepted social rules, norms and social communication capability to solve a given problem collaboratively [124], which is essential for developing socially intelligent autonomous agent.

2.3.6 Autonomic Computing Knowledge Model

Knowledge representation and management are one of the important aspects of developing intelligent systems. The more knowledgeable systems are the closest to achieve a real intelligent system [242]. To determine actions, the autonomic agent relies on its knowledge base, which contains a priori defined policies [209]. The knowledge is the core component in an autonomic manager, which allows access to data shared by the MAPE functions [69, 128]. Typically, a knowledge source is an implementation of a registry, dictionary, database or another repository who provides access to knowledge according to the interfaces prescribed by the architecture [4].

Behavioural knowledge and knowledge execution is a vital research area for the successful fulfilment of Autonomic Communication [230]. The representation of knowledge at the autonomic manager level is a critical issue for designing and deploying self-governing systems [141]. The autonomic computing architectural blueprint [165] outlined three broad types of knowledge: i) solution topology knowledge (represented by installable unit descriptors), ii) policy knowledge (i.e., goals, objectives or actions to be taken), and iii) problem determination knowledge (such as monitored data and symptoms).

Different approaches to knowledge modelling and representation have been developed for intelligent systems. Initially, Knowledge representation utilises multiple techniques, including the incorporation of business rules, decision analytical models and models generated from the application of machine learning algorithms through data mining techniques [218]. Nowadays, the most prevalent knowledge representation models are based on production rules, association rules, frames, logic, decision trees, neural networks, clusters and semantic networks [238, 242, 96]. It is more advantageous to represent knowledge according to the open-world assumption with machine-understandable semantics [5]. Therefore, ontological knowledge modelling is now central to many areas, including communication and knowledge sharing, logic inference and reasoning and knowledge reuse [263].

2.4 Policy-Based Autonomic System

A policy describes principles or strategies for a plan of action that is designed to achieve a particular set of goals [150]. Policy-based computing handles complex system properties and allowing administrators to modify the system's behaviour without changing the source code or reprogramming the systems [109, 198]. Initially, the management of network infrastructure in an enterprise was a complex and daunting affair for trained personnel [244]. Therefore, policy-enabled network management (i.e., security, configuration, recovery, or quality of service) introduced as a flexible solution by which the administration process can be simplified and largely automated to reduced maintenance costs, improved flexibility, verifiability and runtime adaptability [36]. Current policy-based approaches provide a flexible means of specifying adaptation strategies in pervasive systems and sensor networks [35], as well as policy-based data integration for e-Health monitoring [78].

Policy-based management is particularly important with the future vision of autonomic computing, where an administrator may simply specify the business objectives, and the system will operate in terms of the needed ICT [229]. However, an autonomic system requires a well-defined method for defining the policies that will specify decision-making criteria to accomplish certain tasks such as providing services to humans and other autonomic elements or manages and controls themselves [4]. Kephart and Walsh [125] defined three types of policies for autonomic computing based on the notions of states and actions: i) action policies specify what actions should be taken based on the current state of the system, ii) goal policy specifies either a single desired state or one or more criteria that characterize an entire set of desired states and iii) utility function policies are objective functions that express the value of each possible state.

2.4.1 Policy Specification Approaches

There are several approaches introduced to the definition of policies, and accompanying policy languages, which represent different levels of policy expressiveness and semantics, ranging from logic-based languages, special-purpose policy languages, generic rule-based (if-then-else) formats, ontology-based approaches, and so on [36]. Logic-based languages are attractive for the specification of security policy because they have a well-understood formalism that is amenable to machine inference[59]. On the other hand, the rule-based approach is most influential, and most widely-deployed policy language. The classical application of rule-based systems is in expert systems, which typically use a human expert knowledge expressed in terms of rules for solving real-world problems. Other rule-based policy systems adopt an Event-Condition-Action(ECA) rule [22] paradigm. ECA rules are widely used to control the environment as well as to control reconfiguration of software systems. ECA rules have been used in many settings, including active databases for triggering functionality based on data monitoring and workflow management [237], access control and system management to react to different situations [211]. In contrast, the lower-level action-based policies are used to express the actions that should be performed when given conditions are satisfied [4].

ECA rules-based policy specification approach is a well-known and an integral part of the AC knowledge to control the behaviour of the autonomous entities and the interactions with other autonomic elements [211]. ECA rules have been a widely used construct for the high-level specification of controllers in adaptive systems, such as cyber-physical systems and other smart environments, where devices equipped with sensors and actuators are controlled according to a set of rules [43]. Typically, an ECA rule takes the form of ON *Event* IF *Condition* DO *Actions* [22]. The event part describes a situation of interest and dictates when the rule should be triggered, and the action part of a rule is a sequence of one or more events that describes what needs to be done if the condition evaluates true. Executing a rule action may, in turn, may trigger further ECA rules and the rule execution proceeds until no more rules are triggered [189].

2.4.2 Policy or Rule Representation Languages

XML documents are self-describing and provide a platform-independent means to describe data and therefore, can transport data from one platform to another [32]. The major advantage of using XML as a policy representation language is its straightforward extensibility and flexibility [36]. ECA rules on XML data based on the XPath and XQuery standards also use to specify the event, condition and action parts of rules [189]. Ontology languages like OWL are based on logical formalisms and the formal semantics of the language precisely defines the meaning of an ontology in terms of logic [96]. Thus, ontology-based structuring and abstraction help maintain complex sets of policies more expressively [176]. Recently, RuleML [22] has been modularised through a hierarchy of rule sub-languages, encompassing derivation rules (Prolog-like clauses), transformation rules (function definition equations), and reaction rules (Event-condition-action rules) [34]. RuleML is an XML based markup language and formulated based on datalog (a function-free fragment of Horn clause logic) for rule representation in the SW that was proposed by the Rule Markup Initiative.

Policy-based system management deploys complex sets of rules, which may lead to conflicts in the deployed rules due to the dynamic nature of the system. Conflict in policies means that the actions of some rules may override the actions of others due to the overlapping of conditions of some policies. However, policy conflict detection and resolution, along with policy refinement and reasoning, continue to be unsolved areas of policy-based system management [61]. In particular, different kinds of verification techniques perform on the ECA rule-based system, depending on the available information, as described by Cano et al. [43]. According to their study, the first verification to be performed for the detection of inconsistency and syntax errors in the declaration of rules such as the use of undeclared events or unavailable device actions is examples of such errors. Because syntax errors are easily detected by any compiler or interpreter when recognising the ECA rules source code. Additionally, redundancy of rules is detected when the condition and actions of one rule represent a subset of conditions and actions of the other rule. Redundant rules are not directly detected because it can be compiled and executed at the run-time without problems.

2.4.3 Autonomic System Evaluation

Testing an autonomic system and verifying that they behave correctly is still an area of research. However, measuring the reliability and verifying that the autonomic system operates correctly is challenging in large-scale systems, especially when it extends across multiple domains or enterprises [4]. In general, a combination of quantitative and qualitative methods is used to evaluate every characteristic or property of the autonomic system. McCann and Huebscher [110] have suggested a set of metrics and methods such as *Quality of Service (QoS)* to evaluate performance (usually speed or efficiency), *cost of running*, which reduces

[22]http://ruleml.org/index.html

as the system becomes more self-managing, *granularity* for the degree of distribution of the intelligence and so on. Alternatively, the availability of benchmarks, testbeds, and appropriate case studies can help in evaluating and comparing distinct adaptation solutions, at least concerning each adaptation process [201]. On the other hand, testing and validation in a live system are arguably the most accurate way of verifying an autonomic system to ensure its compliance with the set system's goal [79].

2.5 Summary

There are many social collaborative platforms developed that can offer a wide range of features including communication, dynamic collaboration, and forming the community of people. However, these systems are lacks of agent-centric sociality and even not semantically structured to represent and manage cyber-physical relation, collaboration and autonomous decision making. Semantic Web technology can provide knowledge representation and reasoning mechanism for the socio-cyber-physical system. On the other hand, it has been seen from the literature review that autonomic computing can empower socio-cyber-physical collaboration with self-management and adaptive capabilities. An autonomic agent model can be developed and deploy to the system for monitor and manage socio-cyber-physical resources, action coordination, execution and event notification on behalf of these managed resources. Several autonomic computing models proposed to provide solutions in different aspects but not for the self-managed socio-cyber-physical collaboration. More importantly, to achieve a flexible autonomic control, policy model and specification model is required for governing the behaviour of the autonomic adapter during the runtime without re-code whole the system that can minimise the operating cost.

CHAPTER 3

Related Work

This chapter describes the state-of-the-art related work of this thesis that spans into two main research areas: semantic social-collaborative network for cyber-physical resource collaboration and policy-based autonomic computing to manage these resources without human intervention. This chapter discusses and criticises the existing research efforts related to the theory, methodology, framework and tool that were proposed to solve the related problem of this thesis. Finally, this chapter concludes by highlighting the shortcomings of the existing work.

3.1 Semantic Social-Collaborative Network

3.1.1 Definition of Collaboration

Numerous authors have studied the phenomenon of collaboration in different ways and proposed many distinctive definitions and characteristics of collaboration in a variety of contexts. According to Beyerlein and Harris [31], effective collaboration means working together efficiently and effectively to create a successful business function. Petrick [193] suggested that collaboration requires a systems view to solve problems in a complicated environment where agents obtain mutual benefits through their interaction. Ferreira et al. [83] suggested that adequate collaborative software solutions needed to enable individuals and organisations to manage, communicate and work together across time and space barriers. However, before this study, Ellis, Gibbs and Rein [73] shown how the groupware merged computer and communications technology and applied to a broad range of systems. The pointed out that three key areas (communication, collaboration, and coordination) are important to support group interaction. According to Romano et al. [197], collaboration can be established between two researchers (i.e., peer-to-peer interactions) or among groups (i.e., many-to-many interactions). In both cases, it may be implemented by using collaboration tools such as instant messaging, chat, blogs, forums, social networking and so on.

3.1.2 Social Computing for Collaboration

A new computing paradigm termed social computing has gained rapid growth recently to support collaboration and a few research evidenced of a broad range of benefits associated with the use of the social network. Nan et al., [173] defined Social Computing (SC) is the study of computer-mediated communication and interactions among people. They identified that social computing applications like Wikipedia, Facebook and Twitter have met with tremendous success in recent years and revolutionised how people work and communicate with each other. Early studies by Ardini et al., [14] investigated the use of social computing to facilitate online discussions and knowledge sharing, building awareness and supporting team coordination. Chen and Liu [51] explored that social computing and social networks are concerned with the intersection of social behaviour and computing systems, creating or recreating social conventions and social contexts through the use of software and technology. The study by Ardini et al. [14] shown the growing interest in the field of software engineering, mainly in academia, and a general trend toward developing designated social computing platforms and utilising them for management, coding, requirements' engineering, and maintenance and enhancement.

Abbattista et al. [1] presented a survey of social software, as well as investigated the existing tools and environments for collaborative software development. They also discussed some opportunities and challenges of incorporating social software aspects in agile distributed software development. They explored that social software such as wikis and blogs emerged as a practical and economical option, which may use for organising, tracking, and publishing work and sharing knowledge.

A different study by Cordoş, Bolboacă and Drugan [57] emphasised the influence of social media and the importance of patient's social media usage context in retrieving health-related information. However, they pointed out the greatest challenges in harnessing social media due to the constant and rapid evolution, including the continuous development of new technologies and the ever-changing popularity as well as the adoption of specific platforms among different user demographics. Similar social healthcare research by Ventola [243] and Griffiths et al. [95] investigated the benefits, risks and best practices of using social media tools for the future health care delivery using social networks. In contrast, Lapointe et al. [137] examined how individuals and organisations use social media to collaborate to promote health awareness. In the same way, Khorakhun and Bhatti [127] explored online social networks for remote health monitoring.

3.1.2.1 Context-Aware Social Framework

Previously, the use of the social network by collaborative organisations has been used in a variety of contexts, including to create virtual teams, enterprise collaborations, and social movements. There have been significant amounts of research efforts into the modelling of social aspects in the context-aware system. Smailovic and Podobnik [221] described the BeFriend, a context-aware ad-hoc social networking platform based on Facebook and Google+ social graph, which can effectively be used to describe social connections. Lima et al. [144] explored the potential of using context-aware information in Computer Supported Cooperative Work (CSCW) application in order to support collaboration in pervasive environments. They described the approach can be used for the design and development of a context-aware framework that utilises users' context information interpretation for behaviour adaptation of collaborative applications in pervasive communities. Kamberov [121] proposed a solution on social paradigms in mobile context-aware computing in the Smart City context. Its aim to support the dynamical integration of devices into a context-enabled computing system with a predefined structure where the device needs to possess a certain set of information about the surrounding environment and enable cooperation with other elements (i.e., other nearby devices) belonging to a system. Beach et al. [23] offered 'SocialFusion' that capable of systematically integrating diverse mobile, social, and sensing input streams and effectuating the appropriate context-aware output action.

3.1.2.2 Object and Artifacts in Collaboration Task

Several large scale empirical studies discussed how social media artifacts or objects, such as tags, feeds and dashboards, are interrelated to bridge task management in a social environment. Nansen et al. [174] investigated the importance and role of objects in social relations based upon a review of literature, which emphasised that the relationships between humans, material artifacts, and their environments. They cited that the recognition of object sociality, where objects not only materialise relations between people but are also active participants in human social life. Lundberg [153] observed that humans and artifacts are interrelated in collaborative work in order to fulfil some aims or intentions.

Lima et al. [145] investigated 'Software Ecosystem (SECO)' as component repositories that include socio-technical resources. They claimed that social and technical elements need to work together in order to establish and maintain a better network where technical artifacts and actors interact closely. Hazeyama [103] argued that some communications exchanged during software development are closely related to various types of artifacts. He reviewed some developed application like Jazz, Libra_on_Chat, and Bugzilla that support the management of software artifacts (i.e., source codes, UML diagrams, and bug reports, etc.) specific to an aspect in software development. However, he criticised some limitations such as these applications do not support relationships among different types of artifacts or provide a function that associates artifacts with communications.

In contrast, several studies have attempted to identify the sources of collaboration, looking especially at the role of communication and the effects of physical and social objects on the tendency to collaborate. Maamar, Burgeo and Sellami [156] discussed how to design and develop collaborative enterprise applications using business and social artifacts and supports interactions between these artifacts through social software. Begel and Deline [24] introduce 'Codebook', a social networking web service in which people can be friends not only with other people but with the work artifacts they share. The 'Codebook's' graph includes nodes for code at various levels of granularity (DLLs, source files, namespaces, types, members), work items, bugs, revisions, non-code documents, and email messages, as well as people, which enable software engineers to keep track of task dependencies, discover and maintain connections to other teams, and understand the history and rationale behind the code that they work on and use. Marie and Gandon [158] proposed the notion of Object Centred Social Networks that define the union of a social object and corresponding users and interactions that correspond to socially augmented content or representations such as a video or a place. A study by Xiao [257] illustrated the role of physical artifacts to support collaborative work in a healthcare domain. His work suggested that the design and deployment of new healthcare technology should support the functions of physical artifacts in collaborative work by embedding ICT into the existing infrastructure of physical artifacts.

3.1.2.3 Cyber-Physical Object Collaboration

In the past few years, a number of independent research activities that investigate the potentialities of integrating social networking concepts into the Internet of Things (IoT) solutions. The first idea of socialisation between objects has been introduced by Holmquist et al. [104]. An important step in the direction of the SIoT has been accomplished in [134]. There, the implications of the integration between the IoT and the social networks have been investigated and a few interesting exemplary applications are described. That paper, however, does not describe how social relationships should be established by objects and does not propose any solution regarding the required architecture and protocols.

Wei et al. [250] put forward a new concept of 'Physical Objects Social Relationship' for describing, managing, and predicting the relationships between physical objects in IoT. De et al. [62] proposed a conceptual framework for data-oriented Cyber-Physical–Social Systems (CPSSs) and discussed a solution for building human-machine intelligence. Sheth et al. [212] present an emerging paradigm called physical-cyber-social (PCS) computing. It encompasses a holistic treatment of data, information, and knowledge from the PCS worlds to integrate, correlate, interpret, and provide contextually relevant abstractions to humans. They view PCS as the next phase of computing systems, building on current progress in cyber-physical systems, socio-technical systems, and cyber-social systems to support computing for human experience. Weth et al. [246] introduced cyber-physical social networks that took into account both the physical context as well as the virtual context of the user's live and social recommendation service.

3.1.3 Semantic Social Framework

In recent years, various studies have been conducted by researchers to integrate Semantic Web technologies in distributed environments for knowledge sharing and interpretation in the machine-understandable format. Initially, to capture and share knowledge about collaboration, different ontologies have been developed. For example, Oliveira et al. [182] presented a domain ontology and related sub-ontologies for collaboration, cooperation and communication within the context of collaborative web browsing. However, it has a lack of concept to identify a collaboration pattern in different contexts. Bennett and Baclawski [27] cited that wikis one of the most powerful tools for geographically distributed collaboration. They identified the most commonly used Semantic MediaWiki (SMW) that adds semantic features such as tagging and queries to the wiki, but it is relatively limited as an ontology language.

Finin et al. [84] investigated the way the Semantic Web utilised to represent and process social network information. Their finding has shown that FOAF ontology is the most widely used social ontology on the Semantic Web. Golbeck and Rothstein [91] presented an analysis of cross-network linkages with FOAF in many online social networks and explored how this affects network structure, relationships and captures individual behaviour. Challenger [49] introduced ‘UniGrad’ ontology by extending FOAF with graduate education and research domain concepts and designed a social network based on this ontology. Krämer and Conrad [133] proposed a theory of social structures on the Internet that has themed as ontological representation. Chen, Hendry and Huang [53] designed an ontology and integrated with social network data from users and their friends, the user interest and community influences to improve personalised recommendation systems for mobile device users. In an academic context, Obeid et al. [180] proposed an ontology-based recommendation system to assist students to find and select relevant universities and major subject areas in their study of interest. On the other hand, Kumazawa et al. [135] discussed the effectiveness of ontology engineering approach for the process of collaborative research and proposed the way ontology used in interdisciplinary research through the experimental workshops of research development. They described the ontology as a domain-neutral meta-model and illustrated its construction process in a domain-neutral manner.

There are many ontology-based reasoning frameworks developed [6], [224], [239] for supporting cyber-physical system or social networking application in many aspects. Smirnov et al. [222] presented ontology-based information model for smart home devices interaction where the physical level of the cyber-physical-social system described by the ontology. They claimed that the services of physical level linked to the appropriate smart room devices interact in cyberspace while physical devices interact in physical space.

Kabir et al. [119] introduced a ‘Social Context Information Management System (SCIMS)’ with the ability to acquire raw social data from multiple sources. It was an ontology-based model for classifying, inferring and storing social context information such as social relationships and status of entities. Several researchers tend to develop ontology-driven models for health monitoring applications. In the healthcare domain, Lasierra et al. [139] introduced an ontology-driven system to capture knowledge regarding item management and usage for hospitals and medical centres. Chen et al. [52] presented an ontology-based model for diagnosis and treatment of diabetes patients in remote healthcare systems.

Smirnov et al. [222] presented an ontology-based information model for smart home devices interaction where the physical level of a cyber-physical-social system was described by the ontology. Later in another study, Smirnov et al. [224] proposed ontology-Based resource interoperability in a Socio-Cyber-Physical System without agent-oriented functionality. Törsleff et al. [239] suggested an approach for modelling the context of collaborative cyber-physical systems and generated ontologies that can be used at runtime to communicate with each other and perform context-related reasoning. However, this approach did not consider the social aspects of collaborative cyber-physical systems.

Several researchers tend to develop ontology-driven models for health monitoring applications. In the healthcare domain, Lasierra et al. [139] introduced an ontology-driven system to capture knowledge regarding item management and usage for hospitals and medical centres. Chen et al. [52] presented an ontology-based model for diagnosis and treatment of diabetes patients in remote healthcare systems. However, these works suffer from several limitations including the lack of semantic interoperability of the medical connected objects and their data is still poorly represented. Moreover, they are based on patients medical record and clinical practice guidelines without taking into account the emergence of IoT technology [195], the lack of ontology model for representing knowledge of cyber-physical healthcare service in a social context.

3.1.4 Agent-Based Social Framework

Continuous efforts were being devoted by researchers to make information systems as social intelligent systems that encompass human and object interaction with machines. Although some review papers have investigated agent-based modeling and tools for different domain, very few studies have investigated the cyber-physical system design and the applications of semantic ontology for supporting semantic interoperability. Smailovic and Podobnik [221] integrated "Belief-Desire-Intention (BDI)" agents within the "BeFriend" platform that provide autonomous decision-making and enable automated social discovery of new community members.

3.2 Towards Autonomic Computing Approach

3.2.1 Preliminary Autonomic Computing Research

The researcher from both academia and industry has contributed to the design, development, implementation, deployment, operation and evaluation of various architectural aspects and challenges related to autonomic computing approach. After the introduction of initial autonomic computing model by Paul Horn at IBM, in 2001 [106], Ganek and Corbi [89] discussed in details the autonomic computing framework and fundamental characteristics of the self-managed system to deal with increasing software and environment complexity. Then, White et al. [251] described a different architectural approach of autonomic computing to achieve the goals of self-managing components. Afterwards, Sterritt [229] reported the latest autonomic systems research and technologies that influenced the industry. He summarised the state-of-the-art research and technology transfer issues behind autonomic computing approach. The second version of IBM architectural blueprint for autonomic computing published in 2005 [56] that introduced an architectural building block, adoption model, the role of the human in autonomic systems, including delegation of tasks and standardisation. Kephart and Walsh [125] introduced a unified framework based on concepts from *artificial intelligence* such as states, actions, and rational agents. Their framework interrelates three different types of policies (i.e., action, goal and utility function policies) to be used in autonomic computing systems.

Later, the area of autonomic computing research has been moved towards the overcome the issue of different autonomic system modelling problem and its applications in different domain. Tamma et al. [233] developed 'SERSE (SEmantic Routing SystEm)', a distributed multi-agent system that was designed to exhibit autonomic behaviour (i.e., self-configuration and self-healing capabilities) according to events coming from diverse resources. The idea was to provide robust and efficient gathering and aggregation of digital content from diverse resources. Salehie and Tahvildari [201] presented a landscape of research in self-adaptive software by highlighting relevant disciplines and some prominent research projects to identify the underlying research gaps and elaborates on the corresponding challenges. They explored the adaptive properties (known as self-* properties) of IBM perspective of autonomic computing and challenges related to it. Stathis [227] explored an alternative to the original idea of autonomic computing that uses symbolic Artificial Intelligence (AI) agents to support self-governance as an extension of self-management. Belhaj, Belaïd and Mukhtar [26] proposed autonomic systems with learning abilities to render the decision-making processes of an autonomic MAPE-K loop. Zhao et al. [264] presented a survey of autonomic computing and communication in the context of Software-Defined Networking (SDN) and Network Function Virtualisation (NFV) and focused on self-management and optimisation using machine learning techniques. They discuss several goals, research challenges, and development issues on self-management mechanisms and architectures about the software-driven network. Their paper covers multiple perspectives of autonomic communications in software-driven networks, such as automatic testing, integration, and deployment of network functions.

3.2.2 Existing Autonomic System Model

Recently, some of the research efforts in both academia and industry have aimed to tackle adaptation in different levels and domains to maintain autonomic systems functionality. Shuaib, Anthony and Pelc [213] presented an extended intelligent machine design that was used as a basis for defining autonomic computing systems. They pointed out that the autonomic computing system must be certified based on its expected characteristics before it goes live, as these systems have implications from the financial to the space exploration industries. Alaya and Monteil [5] proposed the “FRAMESELF”, a generic autonomic architecture based on the decision models for the self-deployment and self-configuration of machine-to-machine (M2M) systems. FRAMESELF was designed and implemented for self-deployment and self-configuration of M2M communication services according to machines and applications description and environment changes. The Zhao et al. [264] presented a survey of autonomic computing and communication in the context of Software-Defined Networking (SDN) and Network Function Virtualisation (NFV). They further elaborated the issues of automatic service testing, integration and deployment in the context of a virtualised network.

Mezghani et al. [164] proposed a model-driven approach to tackle the complexity of designing autonomic architectures and the related adaptation issues. They considered the remote health care system for in-home patient monitoring as a case study to illustrate the proposed models and their transformations. Recently, autonomic computing reference model superimposed into the areas of context-aware application for sensing, inference and action [178]. Alti et al. [9] described a “Kali-Smart”, autonomic semantic-based context-aware platform based on Semantic Web technologies and a middleware architecture that provided autonomy and reasoning facilities. The main objective of the Kali-Smart platform was to collect contextual data captured directly by sensors, as well as automatically adapting data contents to users.

3.2.3 Social Autonomic System

The integration of autonomic computing approach within social context has drawn the attention of social scientists, academics and enterprise communities for the development of socially-aware systems. Silva et al. [216] suggested that social software users would greatly benefit from the utilisation of autonomic features in a social environment. They argued that autonomic computing could help with the analysis and in decision-making, transforming the network into an intelligent social network. To support the claim, they applied existing autonomic computing techniques to improve social network analysis that primarily concentrated on identifying social entities and their relationships. In contrast, Emilio et al. [74] analysed the self-disclosure problem in social media and provided insights towards a self-adaptive solution. The purpose of that study was to perform constant monitoring over the user sharing activities and notifying when a self-disclosure behaviour detected. Lin, Ho and Lin [147] proposed a framework for developing a Near-Field Communication (NFC) enabled intelligent agent, which combined the NFC technique with context-acquisition, ontology knowledge base, and semantic-adaptation modules to be aware of location, time, device, and activity contexts with respect to personal and social profiles. They also discussed an insight towards a self-adaptive solution, intelligent and social functionalities for the NFC-based intelligent agent (NIA) to support a comprehensive intelligent recommendation and social-cooperation functionalities with appropriate adaptation operators for Social Internet of Things (SIoT) environments.

3.2.4 Cyber-Physical Autonomic System

In the last few years, significant attention has given to employing autonomic computing paradigm for cyber-physical system (CPS) design. Chun et al. [115] focused on applying autonomic computing technologies as an alternative to making a robust and reliable system to deal with uncertainty and uncontrolled condition in the cyber-physical world. To embody autonomous in the software system, they proposed the Autonomic Computing Cycle of CPS (ACCC). Hong et al. [105] proposed an autonomic computing framework for CPS and discussed its overall architecture, development process and operation process at runtime. They implemented the autonomic computing engine and validate the feasibility of the proposed approach through Home Surveillance Robot scenario. Wan and Alagar [247] proposed an agent-based autonomic system architecture that provides global dependable (behaves as expected) functioning of CPS. They proposed Agent-based Autonomic Cover (AAC) to CPS that focused on adaptation, healing, optimisation, and protection of resources. They assigned AAC to monitor CPS sites autonomously and take timely actions to enforce dependable service delivery. Aman and Snekkenes [11] proposed event-driven adaptive security (EDAS) model for Internet-of-Things (IoT) that observe security events (changes) generated by various things in the monitored IoT environment, investigates any intentional or unintentional risks associated with the events and adapts to it autonomously. Recently, Ashraf et al. [16] proposed “Enforced Demand Management (EDM)” in a smart grid as an implementation of the autonomic computing framework, which has primarily focused on the adaptation of the autonomic computing paradigm to make IoT self-sufficient system.

3.3 Knowledge-Driven Autonomic Computing

There has been substantial work in the knowledge-based system in a different context, including autonomic computing, robot control, decision support system and so on. In general, an intelligent system is intended to possess self-awareness capabilities and autonomic behaviour based on well-structured knowledge and algorithms [242]. In autonomic computing era, there is an increasing demand for knowledge model to execute tasks, to make decisions, problem resolution and system collaboration with one another. Sterritt [229] claimed that a vital issue for the success of autonomic computing could be transfer knowledge about the system management and configuration from human experts to the software. He outlined the role of behavioural knowledge in autonomic communication networks, which requires a knowledge layer to facilitate effective, transparent and high-level self-management capabilities and the ability to possess context-awareness. Seleznyov and Hailes [206] discussed a conceptual model for automatic acquisition and processing of knowledge about users and devices in computer networks. Their framework employed autonomous agents for distributed knowledge management and integrated them into an autonomic middleware component. Oliveira, Souza and Perazolo [183] analysed knowledge management for autonomic computing by proposing an environment to help knowledge management of human and autonomic elements. Park, Yoon and Lee [190] described an approach for implementing system knowledge monitoring for the autonomic control system.

Stenmark and Malec [228] shown a generic knowledge-based architecture and representation of knowledge using ontology and discussed a couple of scenarios as possible usage of ontology knowledge in industrial robotic systems. Tenorth, Bartels and Beetz [235] investigated constraint-based motion representations used in robot control that combined with a semantic knowledge base to let a robot reason about their movements and to automatically generate executable motion descriptions that could be adapted to different robots, objects and tools. Tripathi [238] discussed the architecture, knowledge representation techniques and application areas of knowledge-based expert system in Human Resource. Singh et al. [218] presented an “Intelligent Distributed Decision Support Architecture (IDDSA)” that provided knowledge representation and exchange among intelligent agents. Their work provided the basis for the agent-based architecture to support knowledge management, including knowledge creation, knowledge representation, knowledge exchange and the use of knowledge for decision support on a distributed platform. Vassev et al. [242] presented a vision of knowledge representation and awareness in mobile swarm systems formed as open-ended ensembles of special autonomic components.

3.3.1 Policies For Autonomic System Control

For several decades, there have been several works on policy-based systems focusing on developing architecture and specification languages for managing and deploying policies in autonomic computing. Liu and Parashar [149] designed "DIOS++" infrastructure for enabling rule-based autonomic adaptation and control of distributed scientific applications. Mukhija and Glinz [169] proposed "CASA (Contract-based Adaptive Software Architecture)" framework for enabling the development and operation of autonomic applications for dealing with the adaptation concerns at runtime. In "CASA", the adaptation policy of every application has defined externally and specified using an XML-based language, which facilitates changes in the adaptation policy at runtime. Rufus et al. [199] discussed an autonomic system that could allow the system administrator to specify high-level policies to maintain the system without administrator assistance. They developed Jess (Java expert system shell) rule-based policy engine and used Artificial Immune Systems (AISs) to sense environment status and to monitor system components and performance to address the problems of system complexity. Schaeffer-Filho, Lupu and Sloman [204] proposed the "Self-Managed Cell (SMC)" as an architectural pattern for managing autonomous ubiquitous systems comprised of hardware and software components that implemented policy-based adaptation strategies. They have shown that the basic policy-based interactions between management components of autonomous SMCs could be realised through exchanges of notifications and policies. Mola and Bauer [168] proposed collaboration and communication between different autonomic managers at different levels of the hierarchy based on active policies. Bailey, Chadwick and Lemos [21] proposed a "Self-Adaptive Authorization Framework (SAAF)" capable of managing any policy-based distributed "Role-Based Access Control (RBAC)" and "Attribute-Based Access Control (ABAC)" authorisation infrastructure. Calinescu [41] introduced a framework for the formal specification of autonomic computing policies, and used it to define how the autonomic manager at the core of an autonomic system should expose the system to its environment. Anthony [13] presented flexible policy expression language and AGILE framework, which facilitates run-time adaptation policy configuration for autonomic computing.

3.3.2 Policy Specification Approaches and Languages

There are several and most adopted policy specifications languages have been proposed ranging from general-purpose to domain-specific. A popular example is the "Policy Description Language (PDL)" proposed by Lobo et al. [150], which is a simple but expressive language to specify the policy as a function that maps a series of events as a set of actions. The ponder policy specification language proposed by Damianou et al. [59] is a declarative, object-oriented language for specifying security and management policy for distributed object systems. It provides a common means of specifying security policies that map onto various access control implementation mechanisms for firewalls, operating systems, databases and Java. Sloman and Lupu [220] surveyed some of the work on policy specification for both security management and policy-driven network management with an emphasis on practical rather than theoretical approaches. Calinescu [41] introduced a framework for the formal specification of autonomic computing policies, and used it to define how the autonomic manager at the core of an autonomic system should expose the system to its environment.

Rufus et al. [199] developed if-then rule-based policy engine for autonomic system. Anthony [12] presented a policy definition language for the autonomic computing system in which the policies themselves could be modified dynamically and automatically. Anthony [13] presented flexible policy expression language and AGILE framework, which facilitates run-time adaptation policy configuration for autonomic computing. Another widely used rule-based policy system that adopted an Event-Condition-Action (ECA) rule paradigm. In another study, a specification-enhanced ECA model called "Event-Condition-Precondition-Action-Postcondition (ECPAP)" proposed by Shankar et al. [211] for designing adaptation rules. They introduced a new notion called enforcement semantics that claimed guarantees about rule ordering.

Davy [61] used ontologies to create interoperability across different management domains using semantic reasoning, leveraging policy-based management techniques to achieve autonomic behaviour. Quyang [117] presented a policy-based adaptive architecture for pervasive computing and proposed policy ontology and policy language based on Event-Condition-Action rules for well expressive and easily extensible to support the design of the policy.

Bradshaw et al. [36] highlighted the RuleML, which is a generic rule language based on the structure of Horn clauses (Head $< -$ Body), but evolved toward event-condition-action rule formats. One of the most widely-used policy languages for access control is XACML (eXtensible Access Control Markup Language) [255] based on XML, and many implementations for XACML documentation are available on the web site, with varying licensing terms for public download [258]. W3C [1] recommendation of the XML data model is hierarchical and semi-structured [86], which has many advantages to represent the policy for its straightforward extensibility [36] and machine readability.

3.4 Application of Autonomic Computing Approach

Initially, the autonomic computing approach has been successfully applied in many disciplines such as autonomic space exploration missions [240], spacecraft ground systems at NASA [265], scientific collaboration [197] and many other areas. Almomen et.al. [8] examined overcrowding problems in hospitals that threaten the safety of patients who rely on timely emergency treatment. They applied an autonomic computing framework for self-managed emergency departments to regulate and maintain themselves without human intervention. Ayala, Amor and Fuentes [18] focused on the design and implementation challenges of an agent-based "Ambient Assisted Living (AAL)" system that incorporated self-configuration tasks by applying autonomic computing to software agents' internal architecture.

[1]https://www.w3.org/TR/xmlschema-1/

Mulcahy, Huang and Mahgoub [170] proposed an architecture of “Vehicular Ad-hoc Networks (VANETA)” that offered the potential for intelligent transportation networks and claimed that it can actively and passively improve travel efficiency and safety for the vehicles. Kosonen and Ma [131] presented an intelligent traffic signal control method with a combination of autonomic properties both controlling the signals and for modelling the prevailing traffic situation. Aggarwal, Nigam and Shrivastava [3] introduced a new paradigm to reduce the development cost, effort and software maintenance focusing on real-life traffic light management system.

Warnier, Sinderen and Brazier [249] discussed an approach and architecture for a home energy system based on adaptive and self-managing knowledge representation. They applied a weighted rule-based system that adapts continuously to home energy environment. Seydoux et al. [209] proposed “Semantic Internet-Of-Things (SemIoT)” that implements the MAPE-K loop to control the connected devices in the apartment according to the policies fixed by the user.

3.5 Summary

There are numerous study conducted in the areas of semantic social computing and policy-based autonomic computing system and developed many approaches, methodology, tools and framework to support in a various context. However, most of the existing research work only focused on developing systems, whether in the area of social computing or developing autonomic computing system separately and applied in many domains. Previous social computing and networking studies [132], [14], [173] and [171] highlighted the general benefits of using existing Web 2.0 tools or social software for human-centric social networking, communication and collaboration. These approaches successfully applied online healthcare service where the attention paid to create collaborative health awareness using social media [137], remote health monitoring using online social media [127], social media for older adults [142], social networks for future health delivery [95], etc. The limitation of these studies mainly focused on social software in the aspects of social connections, bringing together people and organisation with common interests and allowing them to exchange knowledge and intensify collaboration. Many works proposed theoretical foundation, methodology and framework for collaborative aspects of cyber-physical resources. Very few publications found that addressed the issue of non-human centric social relations, communication and interactions with real-world implementation.

As we see, semantic representations of tasks through ontologies are starting to appear as promising research directions. Several researches [84] and [91] also suggested that modelling of social-collaborative networks can be aided by Semantic Web’s ontologies in a machine-readable way, which is beneficial for data and knowledge integration and reuse of the concepts. These representations enable agents to reason about tasks, for example, to implement activity recognition approaches. However, existing social or other collaboration ontologies has not explored objects or artifact-centric semantic representation in a social-collaborative network.

There has been a significant amount of research efforts into the development of autonomic systems in a variety of contexts, which has partially replaced a few manual managements of system element [251, 149, 233, 169, 265, 42, 209, 266, 26]. A few research suggested that autonomic computing could help with the analysis and in decision-making, transforming the traditional social network into an intelligent social network [122]. Very few works such as [216], [147], and [74] indicated the benefits of integrating social and autonomic perspective. Many studies showed that social networking methodology could be effectively used to represent social connections between cyber-physical resources for developing social context-aware self-adaptive system to support objects collaborative decision making. The integration between the IoT and the social networks have been investigated and a few interesting exemplary applications are described [134]. That paper, however, does not describe how social relationships should be established by objects and does not propose any solution regarding the required architecture and protocols.

There are a lot of works within the greater scope to specify and implement policy-based management and solutions in the autonomic system without fixed coded logic. For example, policy-based autonomic collaboration for cloud management [168], policy-driven self-management in wireless sensor networks [35], monitoring workflows execution using ECA rules [237], ponder policy specification language [59], policy infrastructure for self-managing systems [50]. However, most of the literature suggested the proposed approaches were limited-purpose languages and related to the domain or just implemented a small academic prototype, but not appropriate for driving autonomic behaviour in every possible context.

Overall, based on the above literature review, we can emphasise that numerous researches devoted to the area of autonomic computing and developed many approaches, methodology, tools and framework in different aspects. Comparatively, little attention has been given to the concept of integrating the autonomic computing approach into the cyber-physical-system. Moreover, there are a lot of works within the greater scope to specify and implement policy-based management and solutions in the autonomic system without fixed coded logic. However, most of the literature suggested that existing policy-based approaches were limited-purpose languages and related to the domain or just implemented a small academic prototype, but not appropriate for driving autonomic behaviour in every possible context.

CHAPTER 4

Software Agent-Centric Semantic Social-Collaborative Network

Considerable research has recently focused on integrating cyber-physical systems in a social context. However, several challenges remain concerning appropriate methodologies, frameworks and techniques for supporting socio-cyber-physical collaboration. Existing systems do not recognise how cyber-physical resources can be socially connected so that they interact in collaborative decision making like humans. Furthermore, the lack of semantic representations for heterogeneous cyber-social collaborative networks limits integration, interoperability, and knowledge discovery from their underlying data sources. Semantic Web ontology models can help to overcome this limitation by semantically describing and interconnecting cyber-physical objects and human participants in a social space.

This chapter addresses the establishment of both cyber-physical and human relationships and their interactions within a social-collaborative network. We discuss how nonhuman resources can be represented as socially connected nodes and utilised by software agents. A software agent-centric Semantic Social-Collaborative Network (SSCN) is then presented that provides functionality to represent and manage cyber-physical resources in a social network. It is supported by an extended ontology model for semantically describing human and nonhuman resources and their social interactions. A software agent has been implemented to perform some actions on behalf of the nonhuman resources to achieve cyber-physical collaboration. It is demonstrated within a real-world decision support system, GRiST, (www.egrist.org) used by mental health services in the United Kingdom. The research work presented in this chapter has been published in a few international journals and conferences [114], [113] and [111].

This chapter organised as follows: Section (4.1) discusses the background of the area of the study. Section (4.2) presents a new agent-centric Semantic Social-Collaborative Network (SSCN) model with an extended ontology. Section (4.3) discusses the GRiST mental-health service case study. Section (4.4) implements the extended domain ontology with a Java-based software agent API design. Section (4.5) discuss the evaluation and testing of the ontology with a comparison to related works. The chapter concludes with a discussion and further research directions in Section (4.7).

4.1 Introduction

In the last decade, the emergence of social computing and associated Web 2.0 technologies such as wikis, blogs, and social networks has dramatically changed the way people communicate and share information across communities [53]. Recently, the convergence of cyber-Physical System (CPS) and Social Computing (SC) known as Cyber-Physical Social Systems (CPSS) [46] accelerated construction of smart communities. The fundamental idea is that the various objects (i.e., both human and physical things) can socially interact and cooperate with each other [256]. Within the context of CPSS, it requires tight integration of human-to-human, human-to-object, and object-to-object interactions within the physical, social and virtual worlds [203]. The ultimate goal of a CPSS is to improve the quality of life of citizens in a wide variety of domains such as transportation, healthcare, smart homes, and energy consumption by providing an infrastructure for interactive, intelligent and cost-effective services.

Recent development of the Internet of Things (IoT) seeks to connect computers to objects with self-configuring capabilities for convergence of the physical and cyber worlds [62]. Social IoT (SIoT) is an emerging paradigm of IoT in which heterogeneous IoT devices not only connect and interact together but also socialise and collaborate to achieve a common goal [2]. However, existing research lacks a well-defined methodology and framework for supporting cyber-physical interaction and collaboration in a social space. Most literature on CPSS or SIoT research is primarily focused on low-level developments such as connectivity, platforms, infrastructures, and component/system security.

Gaps remain with developing highly collaborative cyber-physical systems [239]. Although some of the social aspects are partially discussed in the literature, additional efforts are still needed for collaboration among various cyber-physical entities within the operating environment [175]. The challenge is to implement a socio-cyber-physical collaborative system where not only human but also cyber-physical resources (e.g., physical devices, network components, software entities, and digital documents) can interact and communicate socially for collaborative decision making. Developing social relationships between physical objects will facilitate solving problems with multiple physical-object collaborations [250].

In general, social networking empowers and encourages participation, conversation, openness, content creation and socialisation among a community of users [88]. A social networking approach has been successfully applied in many domains such as plant science [186], software engineering research [217, 25], scientific research [197], and collaborative enterprise application [156]. These so-called social networking applications are built around connections between people through their objects of interest such as a blog, forum, tags, bookmarks, images, and groups of friends.

Object-centred sociality [75] was introduced for connecting people via social objects of interest related to their jobs, workplaces, and hobbies. Social objects have a diverse representation including events, multimedia content, people, interests, activities, and things, which provoke, support, maintain interaction, and create social relationships or bonds with at least one other person or a group of people[250, 158]. People do not just connect to each other, they also connect through shared objects and work artifacts in a social network [24]. These relations between documents, software components, technical designs and other work artifacts can influence the collaboration process [148].

Several reference models, frameworks, applications and infrastructures have been proposed that support object-centric sociality such as wikis, groupware, google, and codebook. Unfortunately, these social applications are not adequate for establishing cyber-physical and human interactions within a social context. They do not entirely recognise how objects can establish and maintain their social relationships, how they can be used and understood by people, or how they tie people together within a social network [174]. The fundamental research challenge is to investigate and find a way of representing and managing cyber-physical objects in a social space for building smart communities.

Previously, social agents have been introduced for acting as a representative of users on social networking tasks, such as to track and change users' behaviour, produce content and create connections over existing social networks [10]. A collaboration agenda includes the collaboration's goal, policies and the action selection order enabling the agents to achieve a shared goal [92]. This chapter proposes a framework for software agent-centric sociality that will not only act on behalf of humans but also provide the functionality to represent, manage and perform actions on behalf of cyber-physical resources in a social network. The framework will allow heterogeneous systems and resources to work together in a decentralized manner, creating the foundations for collaborative work between humans and cyber-physical networks.

A cyber-physical object-centric social network must be able to expand dynamically as new and diverse objects are added. This requires an intuitive and meaningful representation for navigating and inferring information from heterogeneous data sources. Semantic Web-based ontologies [30] provides appropriate representation approaches by semantically describing resources and their properties in a social network. An ontology is a knowledge model that explicitly specifies the domain concepts, properties and relationships, where reasoning and inference mechanism can be applied to deduce new information (i.e., discover new relationships) [181].

Core languages for producing knowledge representation models include Resource Description Framework (RDF), RDF schema (RDF(s)), and Web Ontology Language (OWL) [232]. Ontologies using them have been employed in many social applications to describe formal representations of domain knowledge, often in the form of taxonomies [133]. For example, standard social ontologies such as Friend of Friend (FOAF), Semantically-Interlinked Online Communities (SIOC), Simple Knowledge Organization System (SKOS), and Dublin Core (DC) have defined social network structures in many domains [91, 101, 37]. However, they are not capable of semantically representing cyber-physical objects and their interconnection within a social network. More recent efforts [239, 224] have still not provided a comprehensive model of semantic relations, interactions, or reasoning within a socio-cyber-physical context.

This chapter proposes an agent-centric Semantic Social-Collaborative Network (SSCN) that can be built on top of an existing social network. The significant advantage over existing systems is that it allows social connections and collaborations not only between humans but also among cyber-physical resources in a social space with the help of software agent. An extended ontology model is introduced that semantically defines the concepts, properties and complex social relations between human and cyber-physical resources in the SSCN. The proposed approach is implemented and validated by applying it to a large-scale social-collaborative healthcare service called GRiST [1] used within the United Kingdom. The case study demonstrates the online healthcare aspects to create and manage human and nonhuman resources (i.e., patients, care workers, physical devices, etc.) using the software agent. The domain ontology allows semantic interlinking of these resources and the ability to infer when interventions and social collaborations are required to meet healthcare needs. A Java-based Software Agent API is built that provides functionality to create and manage nonhuman resources and evaluate the effectiveness of the proposed collaborative network. This approach helps to maximise the user experiences in collaborative healthcare services such as emergency notification service based on the medical device data and social context.

4.2 The Proposed Agent-Centric Semantic Social-Collaborative Network

This section discusses the design and development of a software agent-centric Semantic Social-Collaborative Network (SSCN) with an extended ontology model. The main idea is to establish social and semantic relations between distributed cyber-physical resources, including humans and their work artifacts, in a social-collaborative environment. The design considerations of the proposed SSCN is motivated by the current limitations of an existing object-centric social networking methodology with regards to supporting cyber-physical resource interactions and collaborations in a social context.

Our framework leverages some existing social networking features (e.g., blogs, tags, bookmarks, sharing, instant messaging) with extended functionality relating to supporting socio cyber-physical collaboration. Additionally, we concentrate on the establishment of semantic social relations and interactions between cyber-physical objects as well as human

[1]https://www.egrist.org

entities, which can be represented and managed by the software agent. We design an extended ontology model to support the task by semantically defining and interlinking the socio-cyber-physical resources and their properties to facilitate data interoperability and infer new information using reasoning rules.

4.2.1 The Design of the SSCN Framework

The proposed SSCN framework is comprised of two core functional subsystems: i) an Extended Social-Collaborative Network (ESCN), which is built on top of an existing Social Network (SN), and ii) an Ontology Service Module (OSM). The *Human Participants* and *Software Agent* are the entities that represent and manage cyber-physical and social resources and work artifacts cooperatively in the ESCN. All these subsystems together form the overall SSCN as shown in Figure 4.1.

Fig. 4.1 Semantic Social-Collaboration Network Framework

The Extended Social-Collaborative Network (ESCN) is a subsystem of the SSCN and comprises of a *Social Network* environment and *Collaboration Workspace (CW)* where both human and software agents are the active participants in the system. The SN environment represents both human and nonhuman entities (i.e., shared social objects, collaborative work artifacts and cyber-physical resources) as network nodes and their properties and relationships are represented as vertices. The SN also allows having an account and social profile page of these nonhuman resources (i.e., detailed information, properties, interactions and social links, etc.) similar to a human social profile. Human entities in the SN can establish and maintain their social status, relations and interactions such as create and update their profile, publish and share information, make friendship-forming groups, etc. whereas nonhuman entities can perform similar tasks with the help of a software agent. An event notification mechanism is important for proactive delivery of event information to focused users or resources and to reduce repeated checks for changes or updates of activities.

The Collaboration Workspace (CW) consists of logical containers, where the human and software agents interact to manage resource activities. The software agent is a program composed of a set of agent templates, defining the structure and behaviour of specific kinds of agents. Each agent acts on behalf of the nonhuman resources and interacts with the SN environment. In CW, the software agent encapsulates functionality to represent, manage and maintain distributed cyber-physical resources (i.e., software elements, physical sensors or devices, network components, etc.) or collaborative work artifact (e.g., digital documents, project entities, task schedule, programs, applications and so on) in the whole ESCN. The agent interacts with the environment through its sensors. Sensors collect event data from the resources and formulate actions based on the deduced information from the ontology and execute through an effector. For example, if any changes or interactions occur in a resource, an event is generated, which is then monitored and captured by the agent. The agent then communicates with the OSM and sends a notification to the relevant SN node based on the inferred information from the ontology.

The Ontology Service Module (OSM) provides ontology management and inference services to the system. An upper ontology links with the *Extended Socio Cyber Ontology* and explicitly models abstract semantic relationships between the nonhuman objects. The extended ontology is also linked with the existing *FOAF* and *SIOC* social ontologies as well as *DC* metadata. The extended ontology is linked with a *Domain Ontology* that borrows necessary terms to represent the ESCN domain knowledge. This whole integrated ontology formally represents the ESCN structure, and it updates the ontology as changes happen in the network environment with the support of the software agent.

The Reasoning Engine is an application or subsystem of the OSM that provides the functionality of consistency checking, concept satisfiability and automated reasoning (i.e., inferring logical consequences from a set of asserted facts or axioms). It provides an interface for working with any existing OWL reasoners such as FaCT++, HermiT, Pellet, and Racer for automated reasoning tasks. The reasoners provide basic reasoning services to apply to OWL ontologies and use description logic-based algorithms for reasoning tasks. The reasoning engine also provides a mechanism to create and manipulate SWRL reasoning rules so that the ontology can make logical inferences based on a set of axioms and input data. The software agent uses this deduced information to perform some actions on the environment or resources.

4.2.2 Ontology Modelling

The proposed ontology model discusses a way to represent socio-cyber-physical collaboration network data in a shared and machine-readable way. The main objective is to capture additional nonhuman related concepts, properties and relations while reducing the unintended modelling costs and human efforts. The main reason for the additional semantic is to overcome limitations of existing ontologies, which are incapable of representing cyber-physical and human sociality. For ease of development, we split our new ontology model into an *UpperOnto* ontology and an extended *SocioCyberOnto* ontology model. A separate domain ontology that reuses concepts and properties from extended ontologies for semantically representing the target domain knowledge is discussed in the implementation section of this chapter.

4.2.2.1 Defining the UpperOnto Ontology

The *UpperOnto* ontology is the main building block that defines a minimal set of abstract concepts and properties to semantically represent cyber-physical resources in the ESCN environment. It is a light-weight ontology that can be reused or extended further to define domain knowledge.

Fig. 4.2 Upper Ontology Abstract Concepts

As shown in Figure 4.2, we distinguish the upper ontology concepts precisely between *umanEntity* and *NonHumanEntity*, which are defined as the subclasses *Owl:Thing.* The class *Owl:Thing* is predefined and every OWL class is a subclass of *Oowl:Thing.* The *HumanEntity* class describes a person's related concepts and the *NonHumanEntity* class describes various types of cyber-physical resources, software elements, social objects and collaborative artifacts in the ESCN system. In the *UpperOnto* ontology, it is not necessary to further conceptualise the *HumanEntity* class as this can be extended by borrowing terms from existing *FOAF* and *SIOC* ontologies, such as the *foaf:Person* class. However, we need to extend the nonhuman entity class to describe the object-centric social relations, which is new to the ontology definition. For instance, the class *NonHumanEntity* is extended, which has a subclsses of *CyberPhysicalObject*, *SoftArtifact* and *SocialObject*. The *SocialObject* class corresponds to the socially augmented objects such as blogs, forums, videos, or a place, with additional deconstructed subclasses including *WorkArtifact*, *Bookmark* and *Tag/Category*. Moreover, the *SoftwareAgent* class represents a group of agents acting on behalf of nonhuman resources, where they can interact and collaborative together to perform certain tasks supporting their activities.

4.2.2.2 The Extended SocioCyberOnto Ontology

The extended *SocioCyberOnto* ontology reuses the concepts and properties from the *UpperOnto* ontology as well as existing *'FOAF*, *SIOC* and *DC* ontologies. The constructed *SocioCyberOnto* ontology, as shown in Figure 4.3, illustrates how cyber-physical objects and artifacts are semantically connected. The extension is limited to a very abstract conceptualisation of cyber-physical object-centric sociality, but it can be shared to develop domain-level ontologies by defining additional concepts and properties to meet the domain-specific knowledge representation.

As shown in Figure 4.3, the extended *SocioCyberOnto* ontology borrows concepts and properties from the *UpperOnto* ontology. For example, the classes *uo:HumanEntity*, *uo:NonHumanEntity*, *uo:SoftwareAgent* and their associated subclasses such as *uo:CyberPhysicalObject* and *uo:SocialObject*. The class *uo:HumanEntity* is linked with the classes *foaf:Person*, *foaf:Organisation* and *sioc:UserGroup*, borrowed from the *FOAF* and *SIOC* ontologies. The nonhuman entities can interact with human as well as social objects such as blog posts and are , therefore, linked by the property relation `interact` with the classes *foaf:Person* and *sioc:Item*. The software manages the nonhuman entities by means of create, share, notify, etc. and links them as *uo:SoftwareAgent* `manages` *uo:NonHumanEntity*.

4.3 Case Study - GRiST Online Healthcare Service

To demonstrate the applicability of the proposed approach, we consider a real-world case study, called GRiST (Galatean Risk and Safety Technology), with the aim of implementing a richer semantic social healthcare network. The case study explores the potential application of our agent-centric SSCN framework. It also illustrates the benefits of the ontological model for representing healthcare domain knowledge with regards to socio cyber-physical healthcare resource interoperability.

Fig. 4.3 Extended SocioCyberOnto Ontology

First, we discuss social connectivity between the human participant and nonhuman medical objects (i.e., heart sensor, digital health record, etc.), represented as nodes in the extended social-collaborative GRiST healthcare system. Second, we discuss the semantic representation of the cyber-physical medical object and social object by constructing an extended domain ontology.

The particular application of GRiST for this case study is for older-adult healthcare, which was a project sponsored by the European Union within its European Institute of Innovation and Technology Health Knowledge Innovation Community (EIT Health) [2]. The GRiST older-adult system provides expertise directly to older adults and their network of carer's such as family, friends, and clinical services to ensure they are safe, secure, and thriving [39].

The cyber-physical resources and care objects or work artifacts across the GRiST healthcare network consisting of a variety of IoT devices, medical assessment and treatment plan documents, digital health record database, and web resources as well as human participants. These shared objects are connected to the patient for monitoring and detecting mental-health and wellbeing conditions. They send alerts to emergency responders or family members and deliver an appropriate treatment plan for the patient. Currently, human participants can interact, communicate and send messages within the care network in different ways for

[2]https://www.egrist.org/grace-age

delivering care services. However, GRiST and many other similar online healthcare systems do not wholly encapsulate social relations, interactions and collaboration for nonhuman medical resources. These healthcare resources and their data are not semantically represented: metadata and relationships are not described by machine-interpretable ontologies and are not able to take advantage of ontology inferencing. Figure 4.4 illustrates the GRiST healthcare service scenario in a social-collaborative perspective with extended social relations involving human and nonhuman entities.

Fig. 4.4 GRiST Social-Collaborative Healthcare Network

4.3.1 Scenario A - Simple Social Networking

Scenario A discusses simple social connections and interactions between human and nonhuman entities in the GRiST healthcare system. In the GRiST social-collaborative network, human entities may include *Patient*, *Doctor*, *Carer*, and *Family*; a few cyber-physical medical resources such as *Heart Sensor*, *Activity Sensor* and *GPSSensor*. The care artifacts may include *DigitalHealthRecord*, *HealthAssessmentDoc* and *PersonalTreatmentPlan*. All of these entities are socially connected as network nodes and they interact and communicate with each other to provide collaborative care support to patients. As shown in Figure 4.4, a *Patient* is socially connected with a *Doctor* by the *allocated-gp* property relation, and with a *Carer* who provides daily care support to the patient by visiting patient's home. The *Patient* is also connected with his *Family* as a friend relation, which is required in case of an emergency contact. A *Doctor* and a *Carer* are also connected because they are co-workers in the same care network, which builds a network of care support. All of these entities and relationships shown in the scenario are maintained by the social networking system with the help of agents, which is easy to implement.

4.3.2 Scenario B - Cyber-Physical Object Sociality

Scenario B also illustrates in more detail the complex social connections and interactions between cyber-physical resources and healthcare artifacts in the GRiST social-collaborative network. Software agents represent and manage these resources, which are created, accessed, and used in a variety of contexts to deliver a distinctive aspect of socialisation and even coordinate and trigger collaborative work in the care network. As shown in Figure 4.4, there may be different kinds of physical sensors attached to a patient's body, including a *HeartSensor* to monitor heart rate, an *ActivitySensor* to measure physical activity, a *GPSSensor* to determine the location and a *SmartDevice* to receive notifications if the person's health is at risk. Similarly, the nonhuman entities can establish social relations and communications based on their status, interests, preferences and goals. For example, different types of physical sensor connected to a single patient can establish social relations with each other through a friend, or by following and notifying each other. As shown in Figure 4.4, a *HeartSensor* can be a friend of an *ActivitySensor* by being connected to the same patient, which means they know each other.

These medical sensors are represented as social network nodes that receive physiological data from the patient and managed by *Agent1*. The sensor data are stored in the ontology and information is deduced by reasoning rules that generate a *HealthStatusReport*, managed by *Agent2*. The *'Agent2'* retrieves the deduced information from the *HealthStatusReport* and if the health condition is at danger level, it notifies the associated doctor's *SmartDevice* based on the additional inference rule. Afterwards, the doctor can view the alert in a smartphone, assess the report and prescribe a treatment plan, which is a kind of document artifact called the *TreatmentPlanDocument*.

Care resources also include the *DigitalHealthRecord* that stores health assessment data, which is managed by *Agent3*. A *HealthAssessmentDocument* related to an individual patient's health assessment is normally created by a carer and stored in the health record database. This is normally accessed by the doctor manually to see the assessment report and suggest treatment accordingly.

In addition, the *http://MedlinePlusWeb* for example, is a web resource that contains health information and other medical artifacts, and *ClinicalGuidance* is the page of the website that describes the various types of health policies and information. These web artifacts may socially be connected to each other as well as with the human user by the relation of friends, follows, storedIn, createdBy, suggests, etc. property relations.

4.4 Implementation

First, we discuss the design and implementation of the GristOnto ontology using an OWL language. The *GristOnto* ontology is built and merged with the *SocioCyberOnto* ontology using the Protégé-5.5.0 ontology editor [3]. Protégé is a free, open-source platform that provides a suite of tools to construct domain models and knowledge-based applications. The resulting ontology is populated with the GRiST synthesised dataset and automated reasoning is performed by launching the FaCT++ ontology reasoner to determine ontology consistency, syntax and semantics. Afterwards, a set of reasoning rules written in SWRL are applied to test the correct operation of the ontology by inferring the medical status of a patient from raw sensor data. SPARQL queries are also performed to determine the correct search operation in the ontology by retrieving health-related information, similar to the structured database search. Secondly, we implement a software agent API using the Java programming language that can provide the functionality to create, manipulate and represent cyber-physical resources in the social-collaborative network system.

4.4.1 Constructing the GRiST Domain Ontology

To design the GRiST care domain ontology, first, we created a mind map that defines the most general concepts in the domain and subsequent specialisation of the concepts. Once we have defined some of the classes, we add properties and features to these classes. Then, we reuse concepts and properties from the extended *SocioCyberOnto* ontology and interlink with the *GristOnto* ontology classes. The ontology extension has been done for three aspects: *human-centric* corresponds to the human entity representation; *artifact-centric* corresponds to the cyber-physical and collaborative augmented content representation; and *social-centric*

[3]https://protege.stanford.edu/products.php#web-protege

corresponds to the socially augmented content representation. Figure 4.5 shows the GRiST domain ontology that illustrates the cyber-social medical resources and semantic relations.

Fig. 4.5 GRiST Domain Ontology

Using the Protégé-5.5.0 ontology editor the *GristOnto* ontology is constructed and merged with the *UpperOnto* ontology and *SocioCyberOnto* ontology with appropriate classes, objects and data properties based on the domain ontology, designed in Figure 4.5. To perform the merging process, the extended *SocioCyberOnto* ontology is imported from the local drive where it was created, and *FOAF*, *SIOC* and *DC* ontologies has been imported from their URL. Only those axioms are copied that are needed as part of the target *GristOnto* ontology. A snapshot of the constructed ontology is shown in Figure 4.6.

4.4.2 Software Agent API Design and Implementation

To develop the software agent API, we produced a class diagram as shown in Figure 4.7 and then implemented it as Java classes, which define the abstract structure and behaviour of an agent. The Agent API also provides a mechanism to create and manipulate SWRL rules in an OWL knowledge base.

The agent function is defined in the abstract *AgentTemplate* class, which can be directly inherited to implement an individual agent by defining concrete agent classes to manage individual resources. For instance *CyberPhysicalAgent* and *SocialAgent* classes implement the abstract methods, defined in the *AgentTemplate* class. The individual agent class contains the

Fig. 4.6 GRiST Ontology Class Hierarchy in Protégé

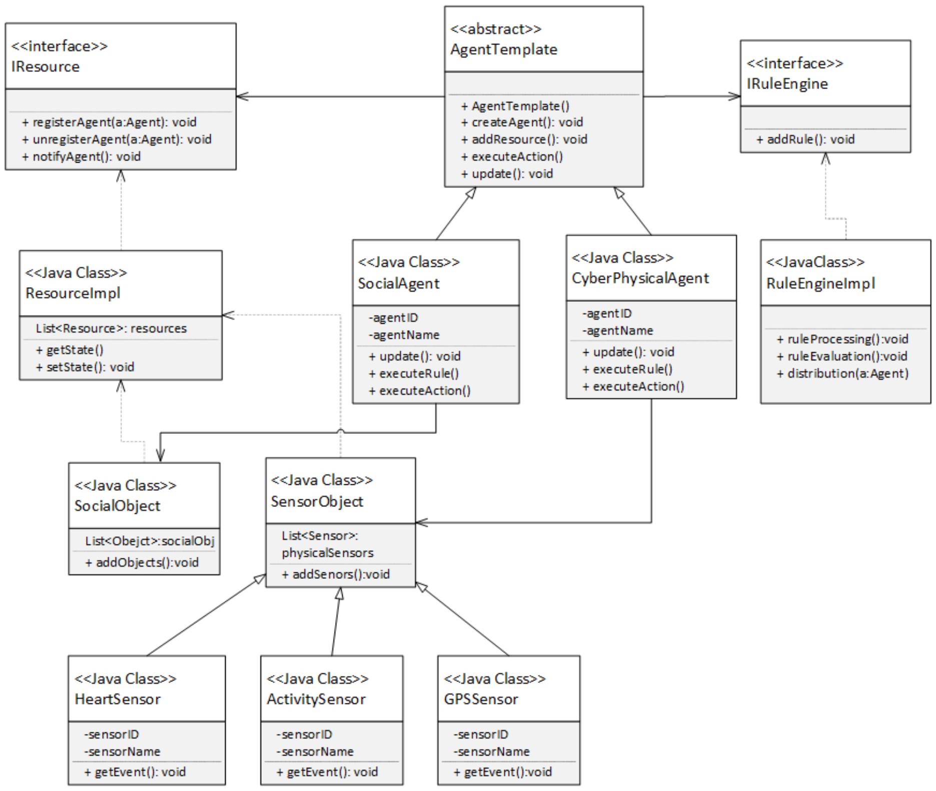


Fig. 4.7 Software Agent API Class Diagram

definition of the functions that the agent has to execute to fulfil its tasks such as observing events and actions to be performed in the environment. Each agent class can be extended to create and manage a specific type of resources. For example, *CyberPhysicalAgent* class is extended to *SensorObject* class, which can also be further extended to create specific types of sensors such as *HeartSensor*, *ActivitySensor* and *GPSSensor*.

The *IResource* interface defines functions to register and unregister agents and an agent notification method. An agent is notified of all the observable events that the associated resource will generate from that moment. All the observable events are detected by the sensor and processed by the agent, which is implemented in *ResourceImpl* class.

The *IRuleEngine* interface provides basic functionality to all rules that inherit from it and the *RuleEngineImpl* class represents a concrete implementation of the *IRuleEngine* interface. The concrete rule engine implements rule processing, evaluation and distribution logic so that an agent can behave accordingly.

To demonstrate programmatically how the agent abstract pattern works, let's create a simple agent abstract class with Java, called *AgentTemplate*. The states and behaviours of the agent are defined in the the *AgentTemplate* by creating a set of variables (i.e., agentID, agentName) and abstract methods (i.e., createAgent(), addResource(), executeAction()) that can be inherited by concrete agent classes. The *RuleEngine* class is responsible for defining and processing rules required to deduce information from the ontology. The code fragment in Listing 4.1 shows the partial implementation of Agent API with Java.

```
public abstract class AgentTemplate {
   public abstract void createAgent();
     // ...
     public final void BuildAgent() {
       createAgent();
     }
   addResourceRule();
   executeAction();
   update();
   // ...
}

public class CyberPhysicalAgent extends AgentTemplate {
   private agentName;
   //constructor
      public CyberPhysicalAgent(String agentID, String agentName) {
         this.agentID = agentID;
         this.agentName = agentName;
          // ...
      }
   //to process action
      executeAction() {
        // ....
      }
   // ...
}

public class SocialAgent extends AgentTemplate {
   //constructor
   public SocialAgent(String agentName) {
      this.agentName = agentName;
      // ...
   }
   // to process action
   executeAction() {
      // ....
   }
   // ...
}
```

Listing 4.1 : Agent API implementation with Java.

4.5 Evaluation

4.5.1 Ontology Testing

To validate the ontological approach, the GristOnto ontology is populated with data synthesised from the GRiST domain. Ontology reasoning and SPARQL queries are performed to test the consistency, correct operation and performance of the ontology. The automated reasoning task using the existing FaCT++ reasoner determines the consistency of the ontology, based on assertion, syntax and semantics. The Semantic Web Rule Language (SWRL) [4] reasoning rules determine whether the ontology accurately derives information from the cyber-physical resource. SWRL allows users to write rules that can be expressed in terms of OWL concepts to provide more powerful deductive reasoning capabilities than OWL alone. Finally, SPARQL queries are performed to determine the correct information returned by the ontology.

4.5.2 Experimental Dataset

The dataset is synthesised from the GRiST healthcare domain as CSV flat documents, which are textual representations of tabular data. In this dataset, the users' related data include patients, doctors, carers and family members with their basic profiles, relationship, explicit interests, and content of social and personal activities. Nonhuman resource data include sensors, social objects, and medical artifacts with their basic information, status, connection, etc.

To enable the dataset for testing, it is generated to match our experimental template for the following information: patient demographic factors (e.g., age, sex, health condition), allocated GPs, associated carers and family members; connected physical devices to monitor patient health condition and location; personal and social circumstances; and healthcare artifacts created by the medical practitioners and software agent. Second, we created the Resource Description Framework (RDF) graph of the generated dataset using the Java-based ontology programming language, known as OWLAPI. The OWLAPI [5] is a Java API and reference implementation for creating, manipulating and serialising OWL Ontologies. RDF is a graph data model that formally describes the semantics or meaning of information.

One of the main advantages of the RDF model is its ability to interconnect resources in an extensible way using a graph-like structure for data (i.e., set of triples, where a triple is represented as subject, predicate, object). The full version of our RDF graph consists of data about 60 users; 15 physical devices that are connected to patients, categorised as heart sensor, activity sensor, and GPS sensor; 10 social objects and 10 medical artifacts created by and collaborated with users, along with their metadata. The size of the RDF graph is sufficient

[4]https://github.com/protegeproject/swrlapi/wiki/SWRLLanguageFAQ
[5]http://owlapi.sourceforge.net/

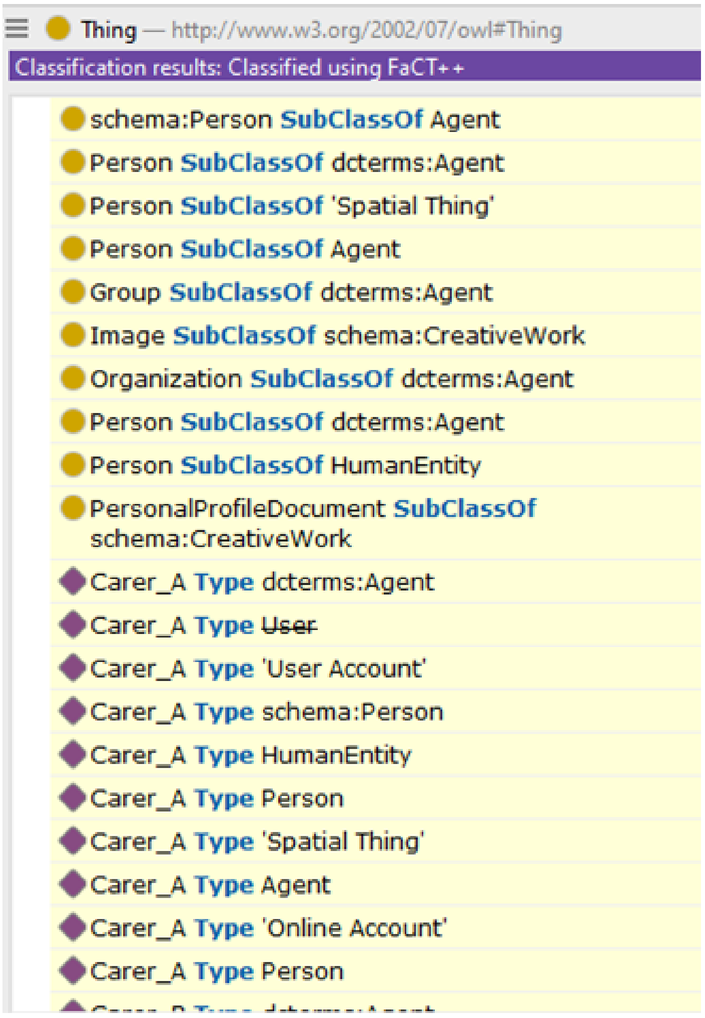
to validate the system in terms of semantic modelling of socio-cyber-physical collaboration. A partial representation of the RDF graph is shown in Figure 4.8.

Fig. 4.8 RDF Model of GRiST Data

4.5.3 Performing Ontology Reasoning and Queries

In this step, ontology reasoning and queries are performed in the Protégé ontology editor to validate the ontology in terms of (i) the consistency and correct syntax of the ontology, (ii) high-risk patient information inferred from sensor data by applying SWRL rules, and (iii) SPARQL queries to verify the correctness of the ontology search.

Ontology Consistency and Syntax. The automated reasoning test is performed on the *GristOnto* ontology by lunching FaCT++ reasoner. The reasoner returns the result as expected, which indicate that the ontology is consistent and error-free as shown in Figure 4.9. If the ontology was not consistent, the reasoner would return an error. The loading of the ontology and import closure is successfully completed in 1544 ms. Classifying the classes and properties and applying the ontology inference was achieved in 110 ms by the FaCT++ inference engine, which is reasonable for this size of knowledge-base.


Thing — http://www.w3.org/2002/07/owl#Thing
Classification results: Classified using FaCT++

- schema:Person **SubClassOf** Agent
- Person **SubClassOf** dcterms:Agent
- Person **SubClassOf** 'Spatial Thing'
- Person **SubClassOf** Agent
- Group **SubClassOf** dcterms:Agent
- Image **SubClassOf** schema:CreativeWork
- Organization **SubClassOf** dcterms:Agent
- Person **SubClassOf** dcterms:Agent
- Person **SubClassOf** HumanEntity
- PersonalProfileDocument **SubClassOf** schema:CreativeWork
- Carer_A **Type** dcterms:Agent
- Carer_A **Type** ~~User~~
- Carer_A **Type** 'User Account'
- Carer_A **Type** schema:Person
- Carer_A **Type** HumanEntity
- Carer_A **Type** Person
- Carer_A **Type** 'Spatial Thing'
- Carer_A **Type** Agent
- Carer_A **Type** 'Online Account'
- Carer_A **Type** Person

Fig. 4.9 Ontology Classification Result Using FaCT++ Reasoner

Ontology Inference Using SWRL Rule. As discussed earlier, the *GristOnto* ontology is constructed to represent knowledge from the GRiST domain that consists of cyber-physical medical resources such as sensors, digital health records, and medical web resources. Different types of physical sensors are connected to patients that measure their physiological data, which is to the social network in real-time. This heterogeneity impedes to perform reasoning across these data sources to generate an integrated view of health report based on physiological and social data from different sources. Let us assume the following example: a domain administrator is interested in inferring the health status of a patient and wants to send an appropriate alert to a relevant person or object (i.e., doctor, family member, or software agent). Consider the following questions: how can an abnormal health condition be automatically distinguished if the measurement of the connected sensor is out of limit; and who should be sent an alert based on varying social situations such as the patient location, nearest people, ambulance, and so on?

Consider the following Description Logic (DL) syntax (1,2, and 3) in Listing 4.2 for the definition of the sensors: *HeartSensor*, *ActivitySensor*, and *GPSSensor* are `subClassOf` *uo:Sensor*. The *HeartSensor* has measurement value 60 bps, *ActivitySensor* has measurement value 1.5 pal and *GPSSensor* has measured value as location home, which are stored in the ontology. The *HealthStatusReport* ontology class represents different levels of health condition that are defined as "normal", "medium" and "high". The *Alert* class represents different levels of notification such as "urgency", "severity" and "certainty". According to the SWRL rule, a reasoner (software agent) is able to infer the patients' health condition from the ontology and notifies the human or other agent via an API call.

```
HeartSensor ⊑ Sensor ⊓ (∃observedData.[hasValue>=60])        [DL Syntax 1]
ActivitySensor ⊑ Sensor ⊓ (∃observedData.[hasValue>=1.5])    [DL Syntax 2]
GPSSensor ⊑ Sensor ⊓ (∃hasLocation.Home)                      [DL Syntax 3]
```

Listing 4.2 : DL Syntax.

For testing the ontology inference, the following SWRL rules are written in SWRLTab, a Protégé plug-in that provides a development environment for working with SWRL rules.

- Rule 1 states that if the heart rate (?h) of a patient (?p) measured by the heart sensor (?s) is between 60 bps to 100 bps, then the heart condition is "normal".

```
Person(?p) ∧ HeartSensor(?s) ∧ hasValue(?s,?value) ∧
    swrlb:greaterThanOrEqual(?value,60) ∧ swrlb:lessThanOrEqual(?value
    ,100) ⟹ hasHeartCondition(?p,normal)     [Rule 1]
```

- Rule 2 states that if the heart rate (?h) of a patient (?p) measured by the heart sensor (?s) is greater than 100 bps, then the heart condition is "too fast".

```
Person(?p) ∧ HeartSensor(?s) ∧ hasValue(?s,?value) ∧ swrlb:greaterThan(?
    value,100) ⟹ hasHeartCondition(?p,too fast)    [Rule 2]
```

- Rule 3 states that if the heart rate (?h) of a patient (?p) measured by the heart sensor (?s) is less than 60 bps, then the heart condition is "too slow".

```
Person(?p) ∧ HeartSensor(?s) ∧ hasValue(?s,?value) ∧ swrlb:lessThan(?
    value,100) ⟹ hasHeartCondition(?p,too slow)    [Rule 3]
```

- Rule 4 states that if the physical activity (?a) of a patient (?p) measured by the activity sensor (?s) is between 1.0 to 1.5, then the activity level is "low active".

```
Person(?p) ∧ ActivitySensor(?s) ∧ hasValue(?s,?value) ∧
    swrlb:greaterThanOrEqual(?value,1.0) ∧ swrlb:lessThanOrEqual(?value
    ,1.5) ⟹ hasActivity(?p,low active)   [Rule 4]
```

- Rule 5 states that if the physical activity (?a) of a patient (?p) measured by the activity sensor (?s) is between 1.6 to 1.9, then the activity level is "active".

```
Person(?p) ∧ ActivitySensor(?s) ∧ hasValue(?s,?value) ∧
    swrlb:greaterThanOrEqual(?value,1.7) ∧ swrlb:lessThanOrEqual(?value
    ,1.9) ⟹ hasActivity(?p,active)   [Rule 5]
```

- Rule 6 states that if the physical activity (?a) of a patient (?p) measured by the activity sensor (?s) is between 2.0 to 2.5, then the activity level is "very active".

```
Person(?p) ∧ ActivitySensor(?s) ∧ hasValue(?s,?value) ∧
    swrlb:greaterThanOrEqual(?value,1.9) ∧ swrlb:lessThanOrEqual(?value
    ,2.5) ⟹ hasActivity(?p,very active)   [Rule 6]
```

- Rule 7 states that if the heart condition (?p, too low) and activity level (?p, low active) and location (?p, home) then alert level is "severity".

```
hasHeartCondition(?p,too low)  ∧ hasActivity(?p, low activity) ∧
    hasLocation(?p,home) ⟹ sendAlert(severity, "do␣some␣exercise") [
    Rule 7]
```

Performing SPARQL Queries To test the correct operation of the ontology and its execution performance, we perform 6 basic SPARQL queries that are frequently issued by practitioners in GRiST domain to retrieve medical information relevant to the patient such as patient details, connected devices, gp, etc. The test data are not for real patients or doctors but are synthesised based on the GRiST knowledge domain.

- Query 1. To find all patient details, the following SPARQL query 4.5.3 is executed, which returns the results of 20 RDF statements, partially shown in Table 4.1.

```
SELECT * WHERE {
?Patient sioc:id ?id .
?Patient dcterms:title ?title .
?Patient foaf:firstName ?firstName .
?Patient foaf:lastName ?lastName .
?Patient foaf:gender ?gender .
?Patient foaf:age ?age .
?Patient dcterms:description ?description . }
```

SPARQL Query 1.

Table 4.1 List of Patients Details from SPARQL Query 1

Patient	id	title	firstName	lastName
:Patient_A	"P001"	"Mr"	"John"	"Dale"
:Patient_B	"P002"	"Mrs"	"Ann"	"Preston"
:Patient_C	"P003"	"Mr"	"Thomas"	"Hartley"
:Patient_D	"P004"	"Mr"	"Sajid"	"Rahman"
:Patient_E	"P005"	"Mrs"	"Mina"	"Katti"
:Patient_F	"P006"	"Mr"	"Mike"	"Daun"
:Patient_G	"P007"	"Mr"	"Xia"	"Feng"

- Query 2. To find patient and doctor relationships where patients are connected to their associated doctors who provide regular consultations and treatments. The following SPARQL query 4.5.3 gives the results of 24 RDF statements, partially shown in Table 4.2.

```
SELECT DISTINCT ?x ?y WHERE { ?x :allocated-gp ?y } ORDER BY ?x
```

SPARQL Query 2.

Table 4.2 Patient-Doctor Relationships from SPARQL Query 2

no	x	y
1	:Patient_A	:Doctor_A
2	:Patient_B	:Doctor_B
3	:Patient_C	:Doctor_B
4	:Patient_D	:Doctor_C
5	:Patient_E	:Doctor_A
6	:Patient_F	:Doctor_D
5	:Patient_G	:Doctor_E

- Query 3. To find the status of all connected sensors for a patient. Different types of sensor are attached to a patient's body that provide physiological data to the GRiST ontology. The following SPARQL query 4.5.3 gives the results of 21 RDF statements, which is partially shown in Table 4.3.

```
SELECT * WHERE {
?Sensor :sensorID ?sensorID .
?Sensor foaf:name ?name .
?Sensor foaf:status ?status .
?Sensor :attachedto ?attachedto .
?Sensor :timestamp ?timestamp .
?Sensor :value ?value .}
```

SPARQL Query 3.

- Query 4. To find the health status of a patient. Data from all connected sensors are combined to generate a health status report using the following SPARQL query 4.5.3 with the results of 3 RDF statements shown in Table 4.4.

Table 4.3 Status of All Connected Sensors from SPARQL Query 3

sesorID	name	status	attachedto	timestamp	value
“hs01”	“Heart Monitor”	“Active”	Patient_A	“12:20”	“96bpm”
“hs02”	“Heart Monitor”	“Active”	Patient_B	“13:11”	“74bpm”
“hs03”	“Heart Monitor”	“Active”	Patient_C	“14:21”	“88bpm”
“as01”	“Activity Monitor”	“Active”	Patient_A	“12:20”	“4.0pal”
“as02”	“Activity Monitor”	“Active”	Patient_B	“13:11”	“2.0pal”
“gps01”	“GPS Sensor”	“Active”	Patient_A	“12:20"	“home”
“gps02”	“GPS Sensor”	“Active”	Patient_B	“13:11"	“street”

```
SELECT  DISTINCT * WHERE {
?x :sdata ?sdata .
?x :related ?y .
?x :riskStatus ?riskStatus .}
```

SPARQL Query 4.

Table 4.4 Health Status Of A Patient from SPARQL Query 4

x	sdata	y	riskStatus
health_report_1	activity_sensor_1	Patient_A	“abnormal”
health_report_1	heart_sensor_1	Patient_A	“abnormal”
health_report_1	gps_sensor_1	Patient_A	“abnormal”

- Query 5. To find the health assessment of a patient from a manual assessment document completed and submitted by a patient or one of the care team to the GRiST system and assessed by a relevant doctor. The following SPARQL query 4.5.3 gives the results of 1 RDF statement as shown in Table 4.5.

```
SELECT * WHERE {
?assessment_document_1 dcterms:description ?description .
?assessment_document_1 dcterms:date ?date .
?assessment_document_1 :createdBy ?createdBy .
?assessment_document_1 :assessedBy ?assessedBy .
?assessment_document_1 :related ?related . }
```

SPARQL Query 5.

Table 4.5 Health Assessment Of A Patient from SPARQL Query 5

assessment_ doc_1	description	createdBy	assessedBy	related
assessment_ doc_1	“The patient Jemmie Lee is suffering from suicide tendency”	“Jemmie Lee”	“Dr. Andrew Garfield”	Patient_A

- Query 6. To find a treatment plan document, which is created by a doctor based on the patient's manual health assessment and health status report generated by sensors. The following SPARQL query 4.5.3 gives the results of 1 RDF statement as shown in Table 4.6.

```
SELECT * WHERE {
?treatment_plan_1 dcterms:description ?description .
?treatment_plan_1 :prescription ?prescription .
?treatment_plan_1 :prescribedBy ?prescribedBy .
?treatment_plan_1 :related ?related . }
```

SPARQL Query 6.

Table 4.6 Treatment Plan For A Patient from SPARQL Query 6

treatment_ plan_1	description	prescription	prescribed By	related
treatment_ plan_1	"Treatment plan for Mr John Dale"	"Patient should do regular exercise and take medication prescribed in document"	"Dr. Fincher David"	Patient_A

4.5.4 Ontology Performance Metrics

To measure the GristOnto ontology performance on query processing, we use the metrics query "execution time" in milliseconds (ms) and the number of "returned results" after processing each SPARQL query, which is shown in Table 4.7. Since we are using only exact queries in a single ontology, the use of traditional Information Retrieval measures, such as precision and recall makes no sense, because the precision will be always 1 in our case.

Table 4.7 Ontology Performance Metrics

Query no.	**Execution time (ms)**	**Returned results**
Query 1	0.4	20
Query 2	0.6	24
Query 3	0.4	21
Query 4	0.2	3
Query 5	0.1	1
Query 6	0.1	1

Based on the ontology performance metrics shown in Table 4.7, we found that the main factor influencing query response time is the number of available axioms in the ontology. The results show that the ontology performs very well for a small-sized RDF statement (i.e., Query 5 and Query 6), but as the RDF size increases the computation cost increases dramatically

(Query 1 and Query 2). This suggests care must be taken in how the knowledge domain is interrogated, making sure the queries are granular enough to be conducted in real-time.

4.5.5 Comparison to Related Works

To assess both characteristics and capabilities of the proposed SSCN system, a systematic comparison between SSCN and most recent related work in the area of study is presented. The key properties that are considered for evaluating the proposed SSCN model are social networking, use of agents, semantic interoperability of the system, and the support for human-centric, object-centric and socio-cyber-physical collaboration. Table 4.8 presents the comparison metrics for different aspects of the system.

Table 4.8 Comparison metrics between the proposed SSCN and related work

Related Work	Social Network-ing	Agent-Oriented	Semantic Interoper-ability	Human-Centric	Object-Centric	Socio-Cyber-Physical Collabo-ration
Alvertis et al. [10]	n	y	n	n	n	n
Schmidt et al. [205]	n	y	y	y	y	n
Törsleff et al. [239]	n	n	y	n	n	y
De et al. [62]	n	n	y	n	y	y
Smirnov et al. [224]	n	n	y	n	y	y
Weth et al. [246]	n	n	y	n	y	n
Kabir et al. [119]	n	n	y	y	y	n
Wei et al. [250]	n	n	y	n	y	y
Nazarenko et al. [175]	n	n	y	n	y	y
Hristoskova et al. [108]	n	n	y	y	y	n
Xia and Ma [256]	y	n	n	y	y	y
SSCN Model	y	y	y	y	y	y

4.6 Discussion

Several studies have explored foundational theory, frameworks, models, and adaptive services across cyber-physical-social systems. They mainly investigated the theories and frameworks for implementing human and object-centric social networks and discussed collaboration aspects between them. Although a few studies addressed the cyber-physical social system, they do not consider how these resources could establish and maintain their social relations and interactions with each other including human entities. In order to assess both characteristics and capabilities of the proposed SSCN system, a systematic comparison between SSCN and the selected 11 most recent related works in the area of study is reviewed and analysed.

Based on our investigation, different socio-cyber-physical or social-collaborative solutions have been proposed for different purposes, each of them presenting advantages and drawbacks. The comparison metrics in Table 4.8 shows that most of the related work is not focusing on developing the mechanism for an agent-oriented approach with semantic social interoperability to support cyber-physical collaboration. Our proposed SSCN model is a comparatively enriched semantic structure for deducing information and performing agent-based actions in the environment.

Additionally, in this chapter, we aimed to model a set of shareable concepts and semantic knowledge of the GRiST domain to demonstrate socio-cyber-physical collaboration in a machine-readable way. A few previous studies developed some well-defined ontological knowledge models for specific cyber-physical domains. However, they are not focusing on ontological knowledge modelling of the agent-oriented approach to managing socio-cyber-physical resources in a social environment. Thus, our specific study goal was to determine whether the Semantic Web ontology was considered suitable for the semantic representation and interoperability between cyber-physical and social systems resources.

Designing and developing ontologies is a complex task, which greatly needs involving appropriate knowledge management methodology and domain experts. On the other hand, there are no globally accepted evaluation mechanisms to measure the ontology performance. Most often, an ontology is tested and validated through performing a series of SPARQL queries and observing the syntax, consistency and correctness of returned results. This similar approach has been applied to test and validate our ontological GRiST knowledge model. Our ontology has passed these tests and demonstrates the benefits of semantic cyber-physical resource interoperability and the feasibility of ontological operations on it within a decision support system. The implemented software Agent API demonstrated how to observe and retrieve events from the resources and perform actions in the environment by manipulating SWRL rules in an OWL knowledge base.

4.7 Conclusion

Current research and development of social software methodologies have no adequate functionalities to support cyber-physical relations and interaction in a social context. Moreover, the lack of semantic definition provides poor data interoperability, accessibility and information discovery from the heterogeneous data sources. To overcome these limitations, we propose the agent-centric SSCN framework. It allows cyber-physical resources to be represented and semantically interconnected as social network nodes, which are managed by the software agent. We develop an extended ontology model that plays a key role in semantically interlinking nonhuman resources in the social world, and supporting ontology reasoning and SPARQL queries for information integration and retrieval.

We discussed a real-world scenario for a collaborative healthcare service context, called GRiST to validate the practical applicability of the proposed approach. The case study demonstrates how SSCN could be utilised to deliver better care support through establishing semantic social collaboration between health practitioners and socio-cyber-physical medical resources (i.e., wearable devices, sensors, care artifacts, etc.). To validate the case study, we developed an extended domain ontology and populated it with test data, synthesised from the GRiST domain, which is then tested and evaluated through the reasoning rules by performing a series of SPARQL queries. The results show that reasoning rules can be beneficial to infer additional knowledge and trigger the appropriate level of alerts by the software agent-based on the patient's physiological medical data and social context.

One of the key limitations of our approach is the inability to handle uncertainty and social dynamics of cyber-physical data due to the complexity and constant changes in resource status as the network grows. Managing and keeping track of these larger numbers of resources is a complex task for human agents even with the help of traditional software agents. Our future work is to develop a knowledge-based intelligent agent model that can autonomously monitor and analyse the environment data and take appropriate actions within the complex socio-cyber-physical system that do not require human intervention. To address the issue, In Chapter 5, we propose a generic autonomic social-collaborative framework and autonomic adapter architecture that is driven by flexible policy.

CHAPTER 5

Generic Autonomic Social-Collaborative Framework (GASCF) and Autonomic Adapter (AA) Architecture

Autonomic Computing (AC) was intended to tackle the growing complexity of Information Technology (IT) infrastructure by making it self-managing and adaptive. The core idea is to make the system intelligent enough to monitor all aspects of the changing environments and its resources, and to control management decisions by defining high-level policies. For several years, great efforts have been devoted to the study of system performance, security, and fault management issues in different IT domains. Less attention paid on the development and adoption of AC theory to achieve the self-management and adaptability within the context of socio-cyber-physical collaborative system. This may be because it is difficult to create such autonomic systems capable to continuously sense and adapt to ongoing changes of cyber-physical resource status and their social context and can make collaborative decision and actions by themselves. The social network plays an essential role in representing the situation in which socio-cyber-physical resources can interact, communicate and collaborate with each other with the help of software agent. However, the remaining challenge for humans or traditional computational agents is to monitor, manage and maintain a large number of socio-cyber-physical resources in real-time as the network grows uncertainly. It also causes additional management and decision making issues, increasing resources, costs and human errors.

To deal with the above problem, this chapter introduce an intelligent agent model, called Autonomic Adapter (AA) driven by the high-level policy. The AAs are integrate into the system for monitoring and managing the socio-cyber-physical resources and to achieve the self-managed and adaptive goal in the deployed system. The AAs are also able to interact and communicate with each other through our proposed Generic Autonomic Social-Collaborative Framework (GASCF). The GASCF is a social-collaborative network built on top of the existing social network. We demonstrate and validate the propose system by applying it to a healthcare system, called eGRiST[1]. The research work presented in this chapter has been published in a international conferences [112].

This chapter is organised as follows: Section (5.1) discusses the background of the area of the study. Section (5.2) discuss the conceptual model of a new Generic Autonomic Social-Collaborative Framework (GASCF). It focuses on a high-level social-context based collaborative system, and the use of AAs makes the system self-managed and adaptive. Section (5.3) introduce AA architecture and section (5.4)the general representation of event-condition-action rule-based policy to control AAs operation. Section (5.5) validate the effectiveness of the approach by applying it to a large-scale collaborative healthcare service, called GRiST (https://www.egrist.org/) that is being used within the English National Health Service and other organisations. Finally, we discuss this work in terms of its overall contribution and what is necessary to evaluate such systems to identify operational performance and effectiveness in Section (5.6). The chapter concludes with a further research challenge in Section (5.7).

5.1 Introduction

Autonomic Computing (AC) was introduced by IBM in 2001 [106] that brings together many fields of computing with the purpose to create self-managed systems [266]. The fundamental design goal of AC was to build an intelligent and self-controlled IT system in the same manner as the human nervous system regulates and protects our body [231]. The theory of AC revolves heavily around the monitor, analyze, plan, execute, and knowledge (MAPE-K) control loop [56]. The initial design AC aimed to decrease some of the complexities associated with the management of the IT system, reduction of maintenance costs and errors, and improvement of services by making the system self-managed and adaptive [227]. Later, AC concept has been extended by the capability of adapting themselves based on context-awareness features [215, 164]. The context-awareness idea was to adjust various artifacts or attributes in response to changes and the context of the operating environment in run-time [201]. At the adaptive level, the system not only monitors, correlate and develops action plans, it also takes corrective actions according to established rules.

Additionally, to be autonomic, a system should have knowledge of itself and its components, as well as the context surrounding its activity, and then be able to act accordingly [106]. This requires an accurate model of knowledge specification, acquisition and processing [202]. For example, rule engines and correlation engines (with their associated languages) are useful

[1]https://www.egrist.org/home

technologies for analysing monitored data and log files to identify trends or situations that warrant deeper examination [125].

Policy-based management using Event-Condition-Action (ECA) rules [22] is a well-known approach and an integral part of the AC knowledge. Typically, an ECA rule takes the form of ON *Event* IF *Condition* DO *Actions* [22]. The event part describes a situation of interest, the conditions determine when the rule can be fired, and the action part executes some change in the environment.

Several AC model proposed that have been successfully applied to tackle the adaptation problem in different levels and domains. For example, AC approach has been successfully applied in autonomic space exploration missions [240], spacecraft ground systems at NASA [265], scientific collaboration [197], autonomic traffic control [231, 131], and self-controlled traffic management [3], enforced demand management (EDM) in a smart grid [16], autonomic communication in the context of software-driven network [264] and so on.

Socio-Cyber-Physical System (CPSS) is a more recent domain where autonomic computing would seem to be particularly appropriate for managing heterogeneous resources. In general, a Cyber-Physical System (CPS) facilitates tight integration between computation, including cyber world (i.e., computers, servers) and the physical processes (i.e., sensors data) through computer networks with the aim to integrate the intelligence in everyday objects/services to execute critical tasks [65]. CPSS integrates various resources from the physical world, cyberspace, and social network [203] that take both the physical context as well as the virtual context of users into account [246]. It can be applied to social services, especially in medical and healthcare applications for real-time monitoring of the patient's health condition and deliver services on the fly. However, having a large number of resources makes CPS unreliable because of the lack of the abilities to change the system behaviour against unexpected changes in the external environment and to adapt for the internal system failures [105]. Therefore, it is an ardours task and error-prone for human or traditional software agents to keep track, manage and maintain these large numbers of unpredicted resource status and their behaviour with the environment in which it is deployed when they are interconnecting in the social-collaborative environment. AC approach has the necessary foundation to tackle this challenge by developing an intelligent agent model with their behaviour governed by the high-level policy. Previous research suggested that autonomic computing approach can be employed to improve collaborative information processing in the social network so that system-wide self-management is achieved [122].

Despite the several architectural models and practical applications of AC in different domain, there are still many technical challenges exists which needs to be addressed to achieve the self-managed and adaptive goals in a social cyber-physical collaboration context. As discussed in Chapter 4, semantic social networking approach can enhance the capabilities for easier collaboration, communication and interactions between human and non-human entities through software agent. However, there are several gaps and dilemmas identified that need to be resolve to achieve self-managed socio-cyber-physical system. The most common issue is to monitor and manage a large number of cyber-physical resources, their social interactions and collaboration using traditional software agent. Within so-called multi-agent systems, agents cannot activate or suppress each other directly but must use accepted social rules and forms of communication to solve a given problem [124].

Furthermore, initial research suggested that an autonomic system design should consider to support communication and interaction cooperatively with various autonomic elements without human involvement [125]. It is a challenging task to establish relations between autonomic managers and their cooperation in social context [168]. Many studies discussed the social intelligence of the agent, which has the ability to interact with other agents (and possibly with humans) via cooperation, coordination, and negotiation rules [253, 67]. The presence of a social intelligence denoted as the sophistication of communicative mechanisms, ability to represent aspects of other agents (individually or grouped), in order to anticipate their actions [70]. Semantic social network is a promising approach to design and implement social intelligence feature within an autonomic agent that can act on behalf of socio-cyber-physical resources to support their social interaction and perform action [113].

To achieve the aforementioned challenge, this chapter proposes Generic Autonomic Social-Collaborative Framework (GASCF) and generic Autonomic Adapter (AA) architecture with their behaviour governed by ECA rule-based policy. The GASCF can be used to implement a self-managed IT system such as social cyber-physical collaborative system, where numerous AAs can be configured and deploy within the IT system to manage resources. Our proposed AA architecture is unique compared to other existing model because it is generic and socially intelligent as they can cooperate and communicate with each other through a social network for collaborative decision making when installing in the target IT system. These AAs uses centralise knowledge repository to decide and perform action on the environment to achieve a shared goal. The AA's *local rule inference engine* can trigger actions based on the monitored data and execute some or all of these actions through its executor after getting confirmation from the *global rule inference engine.* In some cases, an action triggered by an AA may be insufficient to meet operational or system requirements, hence, the *global rule inference engine* outside AA can collate all actions triggered by different AAs and then decides the final intervention according to high-level strategy. The cyfber-physical health care system is one of the popular application which can greatly benefit from the utilisation of our GASCF and AA architecture. We demonstrate and validate the propose system by applying it to the eGRiST[2], mental-health risk and safety management decision support system [71, 40, 38]. The eGRiST is currently used within the English National Health Service (NHS) and other organisations. The research work presented in this chapter has been published in [113].

[2]https://www.egrist.org/home

5.2 High-Level Conceptual Model of the GASCF

The GASCF is a high-level conceptual framework that is composed of multiple AAs that can collaborate and communicate through a Social-Collaborative Network (SCN) for decision making and intervention. A set of managed resources in the operating IT environment are integrated with several AAs, where individual AA can communicate with each other via a social-collaborative network and uses event notification mechanism to activate and coordinate decisions based on the defined policy. The GASCF framework composes of multiple AAs that work in cooperation through the social-collaborative network proposed in Chapter 4. All kinds of connections and interactions between human and resources are maintained through the social-collaborative network, which is monitored and managed by integrated AAs. Figure 5.1 describes the high-level architecture of GASCF, which incorporates three main sub-systems: i) IT Infrastructure, ii) Social-Collaborative Network proposed in Chapter 4, and iii) Autonomic Adapters. All of these sub-systems together create the whole GASCF infrastructure, describes in details below:

Fig. 5.1 High-level Architecture of GASCF

The GASCF can be used to implement self-managed IT systems in a social context to monitor and optimise their resources without human intervention. The example of IT systems may include, cyber-physical social system, distributed network, Social Internet of Things (SIoT), Intelligent Decision Support Systems (DSS), etc. The resources may include all physical devices, cyber components, software elements, social objects, system processes as well as humans who may interact and collaborate through a social network. The event data may generate and come from heterogeneous resources of the IT environment such as the physical devices, software elements, social objects or other artifacts and so on. For example, the state changes of a device (i.e., threshold parameter out of limit), the document created or updated, etc., all of these change to the resources generate events, which is monitor and control by the integrated AAs.

The AAs are a collection of interactive and socially intelligent agents in GASCF, responsible for monitoring, sensing and acquiring raw context data about an event from various distributed cyber-physical resources and then analyse the these data and determines the actions needed for the system. This decision is taken based on the external policies stored in the knowledge repository, configured by domain experts such as sending notifications to resources, update or change something in a resource or communicate with other AAs.

Basically, the social-collaborative network as a whole acts as a communication network between the cyber-physical and social resources, autonomic adapters, human participants and the IT system as a whole. The SCN represents internal or external resources as network nodes and their relationships as edges to support social interactions of these resources. AAs has no direct communications between them but they can communicate through SCN for collaborative decision making. For example, an AA can be represented as a node and connected with other AA as a friend property relation in the social network for sharing event and trigger actions to the target environment or resources.

5.3 Generic AA Architecture

AAs are the fundamental autonomous agents of GASCF that perceive input events from their managed resources through sensors use these percepts to determine actions to be executed through their executors. The resources can be monitored and managed by AAs including physical hardware component (i.e., network components, medical monitoring devices and sensors), a software element (i.e., APIs, a database, a web service), a virtual system (i.e., cloud application, workstation, IT services), any other type of web resources (i.e., social objects, work artifacts), or a IT system as a whole.

The AAs interacts with neighbouring AAs or with the environment as well as humans via a social-collaborative network for cooperation and coordination of decision. Subsequently, AA uses `policies` from central *knowledge Repository*, which is triggered and interpreted as `rules` by the rule inference engine. Typically, an event is generated when there is a significant change occurs in a resource (i.e., failure of CPU, memory overload, database updates, etc.). An action trigger and execute by AA's based on either correlation rules or action rules from the policy to solve allocated tasks by executing dedicated programs. Correlation rules infer a situation by combining several dependent events or actions into one meaningful action. An output action from an AA may raise further input events for another AA during its execution.

As illustrated in AA architecture (Figure 5.2), an individual AA builds upon four main functional elements: a *Sensor Interface (SI)* that manages multiple sensors, an *Intelligent Manager (IM)* loop that consists of additional three sub-elements, including *Monitor (M)*, *Executor (E)*, and a *Local Rule Inference Engine (Local RIE)* that trigger an own intervention. Another important component is *Global Rule Inference Engine (Global RIE)* that collate all actions triggered by AA's Local RIE and make a final decision according to higher-level strategy.

Fig. 5.2 Generic Autonomic Adapter Architecture

i) ***Sensor Interface (SI):*** An AA can manage or control multiple `events` from one or more *managed resources* through its several *sensors* or *listeners*, deeply embedded in the operating system. These sensors are basically software modules, implemented within AA and communicate with resources via web a service interface. Events are generated when there is a significant change occurs in managed resources. Each AA's *Sensor* detects and retrieve the occurrence of the `events` from the variety of resources (i.e., internal or external objects) and then sends the collected `sensor data` to the associated *Monitor* component of an AA.

ii) ***Intelligent Manager (IM):*** An AA's intelligent control loop functionality is configured within the *Intelligent Manager* module, and their behaviour are controlled by the ECA rule-based policy from central knowledge repository. The basic functions of the AA's Intelligent Manager components are discussed below:

- Inside the intelligent manager, the *Monitor* component is responsible to collect, correlate and filter `sensors' data`, received from multiple *Sensor* components. AA's *Monitor* function is to pull out these sensor data and determine which factors are of interest, the priority order and whether this is only for particular value ranges through correlation rules in the policy. In effect, the monitor accesses the policy knowledge and matches sensor data to their associated threshold parameter and determines whether the received data set contains variables and values that match the monitor's threshold parameters in the policy. Next, the monitor sets a new context variable by combining several dependent events into one meaningful event. Afterwards, the monitor communicates and sends the new `variable conditions` to the AA's *Local RIE*, which can trigger some actions depending on these variables.
- *Local RIE* component provides a mechanism to trigger the relevant actions by evaluating the variables given by the *Monitor* component based on action rules from the policy. The local RIE fires the `action`, needed to achieve certain goals and objectives defined in the policy. It is one of the core mechanism that implements the `action rules`, associated with the specific AA that determines what `action` should be fired, if relevant `parameters` are observed, and the `conditions` are fulfilled. The local RIE also determines that if an action is triggered, a message is sent to the *Global RIE* for further action evaluation and correlation, triggered by the other AAs.
- *Executor* component receives the final action command from *Global RIE* and provides the mechanisms to execution of the `action` over the *managed resources* or other AAs via social network environment. To perform this tasks, the executor retrieves the `actions` definition from the policy such as function name, parameters

and values. These actions usually indicate some sort of alteration or adaptation (i.e., setting new variables, triggering a new event, sending an alert or notification message, or do something else) to one or more *managed resource* or sending intervention to a specific social network node.

iii) ***Global RIE:*** A *Global RIE* is a central rule inference engine that implements *Knowledge Repository* and translate policies as independent rules. It is a most essential module for knowledge distribution, where all AA's Local RIE must communicate and send actions. The global RIE is responsible for collating all actions that are active at any one time across all AAs, and model final adaptation plan by scrutinising these actions based on its own higher-level strategies from the knowledge repository. The global RIE evaluates whether all of the action requirements and constraints are met the system requirement and decide if an adaptation is needed or not. If the decision is taken to execute the given AA's intervention, then the global RIE sends the `yes` notification back to the associated AA's *Executor* and the actions are then executed accordingly by calling some relevant functions defined in the policy.

5.3.1 Process Flow of an Autonomic Adapter

The interactions between AA's sensors and intelligent manager's component (monitor, local RIE, executor) uses the event notification mechanism based on the input and output parameters. An event is a type of action or signal to the sensor that something has happened to the objects or the system. As shown in Figure 5.2, each AA's sensor and intelligent manager components have a separate function that takes several inputs and produces outputs according to defined policy. An output action from an AA may raise further input events or interventions for other AAs during their execution. The process flow below describes the high-level input and output data flow between AA's components in real-time.

5.3.1.1 Sensor Process.

Sensors are connected to one or more managed resources and configured to collect specific event data from each managed resource. When an AA's sensors observed data fell within their remit, an event is triggered and these data are then streamed in real-time to associated AA's monitor with which a sensor is connected. The sensors do not by themselves know or care about whether event data is of interest to a connected AA or not. The sensor process is shown in the Process 1.

5.3.1.2 Monitor Process.

The monitor component of an AA is configured to respond to specific elements of the sensor data where the relevance of sensor data is determined. Data may come from multiple sensors

Process 1 : Sensor

```
1: input: generated event $e_i$ from managed resource $mr_i$ or a social-collaborative network
   $scn$ environment that matches with AA's sensor threshold parameter;
2: output: a notification message $n_i$ and a set of sensor data $d_1, d_2, ..., d_n$ to monitor
   component $m$;
3: procedure begin
4:     while at time $t_i$, a new event $e_i$ occurs in $mr_i$ do
5:         if $e_1$ value is within the threshold range fo sensor $s_i$ then
6:             input $e_1$ into the sensor $s_i$;
7:             assign unique key for input event $e_1$ ;
8:             put event $e_1$ into the dispatch queue based on the arrival time;
9:         end if
10:        check next event until 0;
11:    end while
12:    send a notification message $n_i$ to monitor $m$;
13:    send all sensor data $\{(d_1, d_2, ..., d_n) \in S_i\}$ to monitor first-come-first-serve basis
           if requested;
14: end procedure
```

and if the values match the monitor's threshold parameters, a rule is triggered. For each true condition, a new parameter is assigned according to the context requirements. The monitor is then sent a notification message with the new variable set to the local RIE component. The monitor process is shown in the Process 2.

Process 2 : Monitor

```
1: input: a notification message $n_i$ with all sensor data $\{(d_1, d_2, ..., d_n) \in S_i\}$, where each
   monitor input is $s_i \in M$;
2: output: a notification message $n_i$ to Local RIE and a list of assigned context variables
   $var_1, var_2, ..., var_n$ that are returned true;
3: procedure begin
4:     while at time $t_i$, a notification $n_i$ received from sensor $s_i$ do
5:         input all sensor $s_1, s_2, ..., s_n$;
6:         for each input sensor $s_i$ do
7:             if sensor data $d_i$ is equal or within the range of threshold value $v_i$ then
8:                 set a new context variable for each matching condition;
9:                 return true;
10:            end if
11:        end for
12:        send a notification message $n_i$ to Local RIE;
13:        send all context variable $var_1, var_2, ..., var_n$ to Local RIE;
14:    end while
15: end procedure
```

5.3.1.3 Local RIE Process.

The local RIE finds all the rules that have variable conditions matching to the monitor outputs. The local RIE collates and evaluates all new context variable sets. If a condition matched with a particular type of context data, it returns true. For all combined true conditions for a particular group of contexts within an expression, it will trigger actions from the specified policy. The local RIE process is shown in the Process 3.

Process 3 : Local RIE

```
1: input: a notification message $n_i$ with all context variables $var_1, var_2, ..., var_n$ that are
   only returned true;
2: output: trigged action $a_i$ with associated metadata and a notification message $n_i$ to
   Global RIE;
3: procedure begin
4:     while at time $t_i$, a notification $n_i$ received from monitor $m$ do
5:         input all context variable $var_1, var_2, ..., var_n$ that are evaluated to true in monitor;
6:         for a set of expression $Expr_i$ do
7:             collate and evaluate group of context variables $\{(var_1, var_2, ..., var_n) \in Expr_i\}$;
8:             if all context variable within $Expr_i$ evaluated to true then
9:                 return true;
10:            end if
11:        end for
12:        for a set of expression $Expr_i$ do
13:            if $Expr_i$ is true then
14:                trigger actions $a_i$;
15:            end if
16:        end for
17:    end while
18:    send a notification message $n_i$ to Global RIE;
19:    send triggered actions $a_1, a_2, ..., a_n$ to Global RIE;
20: end procedure
```

5.3.1.4 Global RIE Process.

The global RIE collates and evaluates all actions coming from different AA's local RIE for final intervention. This is the core of autonomic adaptation, where the central policies determine how, where and which adaptation takes place. Afterwards, the action plans with command messages are sent to the AAs executors separately. The global RIE process is shown in the Process 4.

5.3.1.5 Executor Process.

The executor component receives *yes* notification with an execution command from *Global RIE*. The execution process carries out by calling a function definition according to policy

Process 4 : Global RIE

```
1: input: a notification message $n_i$ with all triggered actions $a_1, a_2, ..., a_n$ from all AAs;
2: output: an intervention decision with a notification message yes and distribute actions $a_i$ to relevant AA's Executor;
3: procedure begin
4:     while at time $t_i$, a notification $n_i$ received from a Local RIE do
5:         input all actions $\{(a_1, a_2, ..., a_n) \in AA_i\}$ triggered by Local RIE ;
6:         for a set of expression $Expr_i$ do
7:             collate and evaluate group of actions $\{(a_1, a_2, ..., a_n) \in Expr_i\}$;
8:             if all actions within $Expr_i$ evaluated to true then
9:                 return true;
10:            end if
11:        end for
12:        for a set of decision $Dec_i$ do
13:            if $Dec_i$ is true then
14:                confirm actions $a_i$ to be executed;
15:            end if
16:        end for
17:    end while
18:    send a confirmation message $n_i$ to all AAs Executors $\{(exe_1, exe_2, ..., exe_n) \in AA_i\}$;
19:    dispatch actions $a_1, a_2, ..., a_n$ that will be executed;
20: end procedure
```

and then the *Executor* issues the adaptation or sends further action or event to the target resource or the network environment. The executor process is shown in the Process 5.

Process 5 : Executor

```
1: input: a notification message $n_i$ and action execution command $cmd_1, cmd_2, ..., cmd_n$ from Global RIE;
2: output: apply adaptation or action $a_i$ to one or more resource $mr_1, mr_2, ..., mr_n$ or target social network nodes $node_1, node_2, ..., node_n$;
3: procedure begin
4:     while at time $t_i$, a notification $n_i$ received from Global RIE do
5:         for each input command $cmd_i$ do
6:             if triggered action $a_i$ is equal to $a_i \in cmd_i$ then
7:                 call the relevant function or method $f_i$ defined in the policy;
8:             end if
9:         end for
10:    end while
11: end procedure
```

5.4 Policy Specification Language For Autonomic Adapter

In this section, we describe the general-purpose Policy Specification Language (PSL) for AA based on the well-accepted Event-Condition-Action paradigm. The PSL defines an XML grammar for policies, which contains the various elements and attributes of a policy in object-oriented nature. XML is a de facto standard for data representation and interchange, which is convenient for defining the PSL because it is hierarchical.

The PSL is generic, in the sense that is capable of describing a wide range of policies for a very diverse set of application domains. It will be allied to a policy schema for any given domain so that XML documents can be validated against the PSL syntax.

5.4.1 PSL Syntax

The PSL syntax uses the Backus-Naur Form (BNF) [3] notation as follows:

```
<PolicySet>::=<ManagedResource>,
<AdapterID>,{<Policy>};
<Policy>::=<policyID>,<policyName>,<policyType>,
{<Event>,<Condition>,<Actions>};
<Event>::={<eventID>,<source>,<timestamp>,
<description>};
<Condition>::=<var>,<op>,<value>;
<Actions>::={<Action>};
<Action>::=<actionID>,<actionName>,{<Method>};
<Method>::={<MethodSignature>,<Parameter>};
```

- The <*PolicySet*> element is the root element that uses a “ManagedResource” attribute to describe which managed resource belongs to a particular policy set and the “AdapterID” attribute to describe the unique identifier of the attached AA for that managed resource. One <*PolicySet*> element may contain one or more <*Policy*> elements.
- The <*Policy*> element describes the group of policies within a <*PolicySet*>. The <*Policy*> element uses the “policyID” attribute to describe the unique identifier of a policy, the “policyName” attribute to describe the relevant policy name, and the “policyType” attribute is optionally used to describe the policy type of the AA. Each <*Policy*> element is composed of <*Event*>, <*Condition*> and <*Actions*> elements.
- The <*Event*> element contains the attributes “eventID”, “source”, “timestamp” and “description”.

[3] http://cuiwww.unige.ch/db-research/Enseignement/analyseinfo/AboutBNF.html

- The <*Condition*> element contains a "var" attribute to describe the condition variable, an "op" attribute for the arithmetic operator, e.g., "=", "!=", "<", ">", and a "value" attribute to describe the value that is matched to the operator.
- The <*Actions*> element can have one or more more <*Action*> elements.
- Each <*Action*> element represents the action that needs to be invoked for a policy and is specified by the user according to the analysed symptoms (i.e., the matching policy conditions). The <*Action*> element comprises one or more <*Method*> elements. Every individual action must be declared separately using the <*Method*> element inside the <*Action*> element. The <*Action*> element uses an "actionID" attribute to describe the unique identifier of an action and an "actionName" attribute to describe the relevant actions.
- The <*Method*> element defines the concrete action, which contains a <*MethodSignature*> element that represents programming syntax to declare a method signature and a <*Parameter*> element that represents the object or variable instance value to be passed as an object reference. A single <*Method*> element declaration can have one or more <*Parameter*> elements.

5.5 Case Study - GRiST Autonomic Socio-Healthcare System

This section demonstrates the applicability of the proposed approach by applying it to a real-world healthcare service context, called GRiST (Galatean Risk and Safety Technology). In Chapter 4, we demonstrated the GRiST domain within the context of socio-cyber-physical collaboration. In this chapter, we discuss the enhanced functionality of the system by integrating our GASCF and AA approach to make the GRiST system self-managed and adaptive. Integrating AAs into GRiST environment is immensely useful to identify the care environment situations, recognise the resource states and eventually in adapting the system's behaviour according to high-level policy.

5.5.1 GRiST Care Service Scenario

Currently, GRiST system receives data directly from the patients automatically through software sensors that they may be carrying at home environment. Its inbuilt expertise helps the patients to know whether any health or safety problems need to be addressed and also connect them to their care network to elicit help. Due to this facilitation sensor, both the older adults and their carers feel more confident living independently as any problems arise, they can trigger the necessary responses.

In general, the managed resources across the GRiST care domain include a variety of physical devices, software sensors, web resources and network applications as well as human participants. All of them interact, communicate and send messages in different ways for different purposes. For example, one or more wearable devices are attached to a patient's body that records their heart-rate and physical activity level. There are also some software resources or artifacts available within the network environment that controls patients' data and executes specific tasks. Based on the case study, the physical resource includes *Pulse Monitor* and *Activity Sensor* and sends pulse data (bpm) and physical activity level data (pal) respectively to the GRiST system and communicates via a web service (i.e., RESTful API) interface. On the other hand, the managed resource *Assessment Document* associated with some software components that handle the risk assessment document submitted to the network by a patient or practitioners. AAs are deployed and attached to these individual resources that continuously monitor their status within care network. *AA1* manages *Pulse Monitor* and *AA2* manages *Activity Sensor*, and *AA3* links to a software artifact, labelled as *Assessment Document.* The social objects or artifacts in the care network, include *Pulse Node*, *Activity Node* and *Assessment Node* that represent *AA1*, *AA2* and *AA3* respectively, which maintain objects' social connections, communications and interactions. Figure 5.3 shows the deployment of the AAs into GRiST Care Network.

The threshold parameters are set by doctors in the monitor part of each AAs policy separately for certain bio-data types. For example, lower range and higher range of heart-rate bits per minute (bpm) and physical activity level (pal) to measure the intensity of body movement each minute. Moreover, the parameter can be defined to monitor the status of any software artifacts, including new a health assessment document created and submitted to the system, or health database updated, etc. If the associated resource's event parameter is out of range, or any other type condition defined in the policy, it triggers some rule that may depict an action to be sent to the connected nodes in the care network. For example, if the heart rate exceeds or drops below a threshold or no physical activity of a patient, then a doctor, caregiver and a family member could be alerted to contact the patient urgently as illustrated in Figure 4.4.

5.5.2 Prototype Implementation

This section discusses a prototype implementation of our Autonomic Adapter architecture and the general XML representation of policy for the associated AAs, which are configured according to the GRiST case study discussed above.

As shown in Table 5.1, each AA can have their own integrated sensors *S1* and *S2* to observe events from integrated managed resource and GRiST care network nodes respectively. *AA1's* sensor *S1* is attached with the physical managed resource *'Pulse Monitor'* and sensor *S2* is attached with the *'Pulse Node'* in the care network. Similarly, *AA2's* sensor *S1* is attached with the physical managed resource *Activity Sensor* and sensor *S2* is attached with *Activity Node* in the care network. *AA3's* sensor *S1* is attached with the software artifacts

Fig. 5.3 GRiST Autonomic Socio-Healthcare System

(or managed resource) *'Assessment Document'* and sensor *S2* is attached with *'Assessment Node'* in the care network. The *'Pulse Node'*, *'Activity Node'* and *'Assessment Node'* are the network objects in the care network that represents *AA1*, *AA2* and *AA3* respectively, which maintains their social relationship, communication and collaboration.

Table 5.1 Deployment of AAs into GRiST System

Adapters	**AAs Sensors**	**Managed Resources**	**Resource Type**
AA1	S1	Pulse Monitor Device	Physical Resource
	S2	Pulse Node	Social Network Object
AA2	S1	Physical Activity Sensor Device	Physical Resource
	S2	Activity Node	Social Network Object
AA3	S1	Assessment Document	Software Artifact
	S2	Assessment Node	Social Network Object

5.5.2.1 AAs Policy Specification

As shown in Table 5.2, the physical *Pulse Monitor* device is out of the threshold limit “140 bpm” and its value is within the range of interest of the associated sensor, and then the sensor sends retrieved data to monitor of *AA1*. The AA1’s *Monitor* component filters out the received events that are relevant to its remit and accessing the shared knowledge in *Global RIE*, creates a symptom, such as the patient’s heart condition is “abnormal”. The *Monitor* then sends the new condition variable to the *Local RIE*. The rule *R1* gets triggered when the AA1’s *Local RIE* determines that the heart rate is less than “40 bpm” or greater than “140 bpm”. The fired rule triggers a notification actions that tells patient to sit down and breath slowly and steadily until the pulse reduces. It also notifies *AA2* that manages physical *Activity Sensor* device. The *Local RIE* is then communicates with the *Global RIE* and sends the triggered actions to it for further evaluation. The *Global RIE* gets the actions and checks the shared knowledge, whether any other conditions or AA’s input need to be verified before executing any actions. In this case, *Global RIE* rule *R1* get triggered because heart-rate is abnormal and dispatches actions back to the *AA1’s* executor. The *Executor* component calls the appropriate function definition by accessing shared knowledge and sends the executed actions to the *Pulse Node* associated with *AA1* in the network. The *AA1* then distributes the notification messages directly to the person node in the care network and related one notifies *Activity Node* that is listened by *AA2’s* sensor *S2*, which manages physical *Activity Sensor*. This means a message is directly sent to the patient node in the care network. It also causes *AA2* to be activated upon receiving an event notification from the *AA1*.

AA2 receives the notification that triggers rule *R2*, which combines with the *Activity Sensor* data. If the *AA2’s* sensor *S1* detects physical activity output is within its remit, the data is sent to the monitor. The *Monitor* component filter out the event and creates a symptom such as patient’s physical activity is “low” by accessing the shared knowledge and send the condition variable to *Local RIE*. The *AA2’s* rule inference engine checks whether the condition ascertains a physical activity level is less than "1.40" pal in the policy and triggers some actions. The actions are then sent to the *Global RIE* that triggers rule *R2*, which combines the output of the triggered rules from *AA1* and *AA2* and checks with its higher-level strategy such as if the heart-rate is "abnormal" and physical activity is "low", and then sends the actions back to the *AA2*. The *Executor* retrieves the function definition from the shared knowledge and after execution, it sends the actions to *Activity Node* within care network, which distribute the notifications to a doctor suggesting the patient is having a possible anxiety attack, because the pulse is high but activity is low. The same rule also triggers an alert asking the patient to do a manual assessment and also notify *AA2* that manages social object, called *Assessment Document*.

Finally, the *AA3* may have been triggered for a new assessment for the older adult that falls within the time span of the anxiety attack. The AA3 rule *R3* triggers upon detecting an event new assessment document submission from the managed resource *AssessmentDocument* and based on the risk assessment report. If the risk assessment is "normal", subsequently the fired rule trigger actions to notify a doctor, patient and family member confirming that it is an anxiety attack. However, if the risk assessment is "high", then the rule *R4* fired that trigger actions to notify doctor, patient and family member with different alert messages. In *Global RIE*, the output actions from *AA3* combined with the outputs of the triggered actions from *AA1* and *AA2* that determines whether the care network should be alerted for the patient's health condition or risk status. From the global policy, rule *R3* determines that if the hear-rate is "abnormal" and physical activity level is "low" and risk assessment is "high", which then sends the actions back to the *AA3* to execute the actions trigged by the *AA3* rule *R4*. If global policy rule *R4* determines that the hear-rate is "abnormal" and physical activity level is "low" but risk assessment is "normal", then send the actions back to the *AA3* to execute the actions trigged by the *AA3* rule *R3*. The fundamental decision making policies for individual AA's based on the GRiST scenario is presented in Table 5.2.

5.5.2.2 ECA Rule Pattern of AAs Policies

AAs policy specification in the ECA rule format according to GRiST context presented in object-oriented nature as shown below. The object-oriented nature of the policy specification enables highly expressive policy logic using a simple and consistent syntax. In particular, it promotes easy understanding and coding into the programming language and reuse of policy objects significant savings in the time and cost associated with policy development and testing. The object-oriented approach will allow policy attributes to be assigned to the various classes and objects (i.e., rules, events, conditions, actions, variables, etc.) during the program implementation.

- **AA1 Policy** specification in ECA rule pattern shown in below:

Table 5.2 GRiST Basic Policy Specification

Policy Type	**Rule ID**	**Event**	**Condition**	**Action**
AA1	R1	On pulse out of range	If (heartRate < 40 bpm OR heartRate > 140 bpm)	action 1: notify(patient, “sits down and breathes slowly and steadily until the pulse normal”) action 2 : notify(activity sensor, event)
AA2	R2	On receive notification from AA1	If (physicalActivity < 1.40 pal)	action 3: alert(patient, “do a manual assessment”) action 4: alert(doctor, “possible anxiety attack”)
AA3	R3	On received doc notification	If (riskAssessment ==“normal”)	action 5: notify(doctor, “confirmed anxiety attack”) action 6: notify(patient, “confirmed anxiety attack”) action 7: notify(family member, “confirmed anxiety attack")
	R4	On receive notification from AA2	If (riskAssessment == “high”)	action 8: alert(doctor, “patient’s need urgent care”) action 9: alert(patient, “you are at risk, emergency service is on the way”) action 10: alert(family, “immediately give first aid”)
Global Policy	R1		IF (heartRate == “abnormal”)	dispatch action 1 and 2 to AA1
	R2		IF (heartRate == “abnormal” AND physicalActivity == “low”)	dispatch action 3 and 4 to AA2
	R3		IF (heartRate == “abnormal” AND physicalActivity == “low” AND riskAssessment = “high”)	dispatch action 8, 9 and 10 to AA3
	R4		IF (heartRate == “abnormal” AND physicalActivity == “low” AND riskAssessment == “normal”)	dispatch action 5, 6 and 7 to AA3

```
ON (pulseChange(ManagedResource pulseSensor))
  IF ((pulseMonitor.heartRate < 40 bpm) OR (pulseMonitor.heartRate > 140
      bpm)) THEN
    action a1: alert(PatientNode nodeID, msg("sits down and breathes slowly
    and steadily until the pulse normal")
    action a2: notify(ManagedResource activitySensor, Event e1)
  END IF
END
```

- **AA2 Policy** specification in ECA rule pattern shown in below:

```
ON (getNotified(ManagedResource activitySensor))
  IF (activityMonitor.physicalActivity > 1.40 pal) THEN
    action a3: alert(PatientNode nodeID, msg("do a manual assessment"))
    action a4: alert(DoctorNode nodeID, msg("possible anxiety attack"))
  ENDIF
END
```

- **AA3 Policy** specification in ECA rule pattern shown in below:

```
ON (assessmentSubmitted(ManagedResource assessmentMonitor))
  IF (assessmentDocument.riskAssessment == "normal") THEN
    action a5: notify(DoctorNode nodeID, msg("confirmed anxiety attack"))
    action a6: notify(PatientNode nodeID, msg("confirmed anxiety attack"))
    action a7: notify(FamilyNode nodeID, msg("confirmed anxiety attack"))
  ENDIF
  ELSE IF (assessmentDocument.riskAssessment == "high") THEN
    action a8: alert(CarerNode nodeID, msg("patient's need urgent care"))
    action a9: alert(PatientNode nodeID, msg("you are at risk, emergency service
      is on the way"))
    action a10: alert(FamilyNode nodeID, msg("immediately give first aid"))
  ENDIF
  ELSE
    Do Nothing
  ENDIF
END
```

- **Global Policy** specification in ECA rule pattern shown in below:

```
ON (triggeredActions)
  IF (aa1.heartRate=="abnormal")
    dispatch(Action action[a1],[a2]; Adapter aa1)
  ENDIF
  ELSE IF (aa1.heartRate=="abnormal" AND aa2.physicalActivity=="high")
    dispatch(Action action[a3],[a4]; Adapter aa2)
  ENDIF
  ELSE IF (aa1.heartRate=="abnormal" AND aa2.physicalActivity=="low" AND
       aa3.riskAssessment="high")
    dispatch(Action action[a8],[a9],[a10]; Adapter aa3)
  ENDIF
  ELSE IF (aa1.heartRate=="abnormal" AND aa2.physicalActivity=="high" AND
      aa3.riskAssessment="normal")
    dispatch(Action action[a5],[a6],[a7]); Adapter aa3)
  ENDIF
  ELSE
```

```
      Do Nothing
    ENDIF
  END
```

5.5.2.3 XML Configuration of AAs Policies

In practice, policy scripts are formatted in XML with an enforced ECA rule outlined in above section. The XML representation of the policies for AA1 and AA2 is partially shown below, and the full configurations of AA3 and Global Policy scripts are shown in Appendix A.

AA1 Policy Script represented in XML format, which has shown in the Listing 5.1.

```
<AAPolicy aaID="aa1" managedResource="pulseMonitor" policyType="pulse
␣monitor␣policy">
  <Rule ruleID="r1">
    <Event eventID="e1" eventSource="pulseMonitor" event="pulseChange"/>
    <Condition>
      <Expression exprID="ex1" varName="heartRate" op="&lt;" value="40"/>
      <Expression exprID="ex2" varName="heartRate" op="&gt;" value="140"/>
        <Action actionID="a1" actionName="send-alert">
          <Method name="alert">
            <Parameter paramName="patientNode" value="10"/>
            <Parameter paramName="message" value="sits␣down␣and␣breathes␣slowly
              ␣and␣steadily␣until␣the␣pulse␣normal"/>
          </Method>
        </Action>
        <Action actionID="a2" actionName="send-notification">
          <Method name="notify">
            <Parameter paramName="ManagedResource" value="activitySensor"/>
            <Parameter paramName="Event" value="e1"/>
          </Method>
        </Action>
    </Condition>
  </Rule>
</AAPolicy>
```

Listing 5.1 : AA1 policy script in XML format.

AA2 Policy Script represented in XML format, which is shown in the Listing 5.2.

```
<AAPolicy aaID="aa2" managedResource="activitySensor" policyType="activity␣
    sensor␣policy">
  <Rule ruleID="r2">
    <Event eventID="e2" eventSource="activitySensor" event="getNotified"/>
    <Condition>
      <Expression varName="physicalActivity" op="&lt;" value="1.4"/>
         <Action actionID="a3" actionName="send-alert">
           <Method name="alert">
             <Parameter paramName="patientNode" value="10"/>
             <Parameter paramName="message" value="do␣a␣manual␣assessment"/>
           </Method>
         </Action>
```

```
        <Action actionID="a4" actionName="send-alert">
          <Method name="alert">
            <Parameter paramName="doctorNode" value="3"/>
            <Parameter paramName="message" value="possible␣anxiety␣attack"/>
          </Method>
        </Action>
    </Condition>
  </Rule>
</AAPolicy>
```

Listing 5.2 : AA2 policy script in XML format.

AA3 Policy Script represented in XML format, which is shown in the Listing 5.3.

```
<AAPolicy aaID="aa3" managedResource="assessmentMonitor" policyType="assessment
    ␣monitor␣policy">
   <Rule ruleID="r3">
      <Event eventID="e3" eventSource="assessmentMonitor" event="
          assessmentSubmitted" />
      <Condition>
         <Expression varName="riskAssessment" op="==" value="normal" />
            <Action actionID="a5" actionName="send-notification">
               <Method name="notify">
                  <Parameter paramName="doctorNode" value="3" />
                  <Parameter paramName="message" value="confirmed␣anxiety␣
                      attack" />
               </Method>
            </Action>
            <Action actionID="a6" actionName="send-notification">
               <Method name="notify">
                  <Parameter paramName="patientNode" value="10" />
                  <Parameter paramName="message" value="confirmed␣anxiety␣
                      attack" />
               </Method>
            </Action>
           <Action actionID="a7" actionName="send-notification">
               <Method name="notify">
                  <Parameter paramName="familyNode" value="5" />
                  <Parameter paramName="message" value="confirmed␣anxiety␣
                      attack" />
               </Method>
           </Action>
        </Condition>
    </Rule>

 <Rule ruleID="r4">
     <Event eventID="e3" eventSource="assessmentMonitor" event="
         assessmentSubmitted" />
     <Condition>
        <Expression varName="riskAssessment" op="==" value="high" />
          <Action actionID="a8" actionName="send-alert">
             <Method name="alert">
                 <Parameter paramName="carerNode" value="2" />
                 <Parameter paramName="message" value="patient's␣need␣urgent␣
                     care" />
```

```
                </Method>
                </Action>
             <Action actionID="a9" actionName="send-alert">
                <Method name="alert">
                   <Parameter paramName="patientNode" value="10" />
                   <Parameter paramName="message" value="you␣are␣at␣risk,␣
                       emergency␣service␣is␣on␣the␣way" />
                </Method>
             </Action>
             <Action actionID="a10" actionName="send-alert">
                <Method name="alert">
                   <Parameter paramName="familyNode" value="5" />
                   <Parameter paramName="message" value="immediately␣give␣first␣
                       aid" />
                </Method>
             </Action>
         </Condition>
  </Rule>
</AAPolicy>
```

Listing 5.3 : AA3 policy script in XML format.

Global Policy Script represented in XML format, which is shown in the Listing 5.4.

```
<GlobalPolicy>
<InputAction>
<Action id="a1" ref_aa="aa1" />
<Action id="a2" ref_aa="aa1" />
<Action id="a3" ref_aa="aa2" />
<Action id="a4" ref_aa="aa2" />
<Action id="a5" ref_aa="aa2" />
<Action id="a6" ref_aa="aa2" />
<Action id="a7" ref_aa="aa2" />
<Action id="a8" ref_aa="aa2" />
<Action id="a9" ref_aa="aa2" />
<Action id="a10" ref_aa="aa2" />
</InputAction>
<RuleSet>
<Rule ruleID="r5">
<Condition>
<Expression exprID="ex1" varName="heartRate" op="==" value="abnormal" />
<Dispatch target="aa1">
<Action> a1 </Action>
<Action> a2 </Action>
</Dispatch>
</Condition>
</Rule>
<Rule ruleID="r6">
<Condition>
<Expression exprID="ex1" varName="heartRate" op="==" value="abnormal" />
<Expression exprID="ex2" varName="physicalActivity" op="==" value="high" />
<Dispatch target="aa2">
<Action> a3 </Action>
<Action> a4 </Action>
```

```
</Dispatch>
</Condition>
</Rule>
<Rule ruleID="r7">
<Condition>
<Expression exprID="ex1" varName="heartRate" op="==" value="abnormal" />
<Expression exprID="ex2" varName="physicalActivity" op="==" value="low" />
<Expression exprID="ex2" varName="riskAssessment" op="==" value="high" />
<Dispatch target="aa3">
<Action> a8 </Action>
<Action> a9 </Action>
<Action> a10 </Action>
</Dispatch>
</Condition>
</Rule>
<Rule ruleID="r8">
<Condition>
<Expression exprID="ex1" varName="heartRate" op="==" value="abnormal" />
<Expression exprID="ex2" varName="physicalActivity" op="==" value="high" />
<Expression exprID="ex2" varName="riskAssessment" op="==" value="normal" />
<Dispatch target="aa3">
<Action> a5 </Action>
<Action> a6 </Action>
<Action> a7 </Action>
</Dispatch>
</Condition>
</Rule>
</RuleSet>
</GlobalPolicy>
```

Listing 5.4 : Global policy script in XML format.

5.6 Discussion

Autonomic management of distributed IT systems has become increasingly necessary, since frequent manual configuration and fault-diagnosis is error-prone and not feasible in the longer-term [124]. Our research contribution in this chapter focuses on designing a generic autonomic framework and its autonomic adapter architecture driven by high-level policy. Our proposed framework is a modest contribution to the ongoing discussions about developing a self-managed socio-cyber-physical collaborative system, which is easy to implement and universal enough to be applied in most similar cases. We have illustrated that the self-managed properties can be achieved through integrating AAs into the target system, where resource states can be monitored and automatically transforms social context information into decision support based on high-level policies defined by the domain expert. One of the core feature is that AAs can communicate or activate each other through social-collaborative network channel in order to make an collaborative intervention plan, which is not possible using traditional multi-agent system.

Significantly, our AA model has several significant advantages over existing approaches to context-aware autonomic system development. First, it is generic and socially intelligent, because it can be re-configured for use with any legacy or future IT system whose behaviour can be triggered without disrupting the system operation, and they can socially interact with other for collaborative decision. This system consists of multiple autonomic adapters that can be scaled to any degree, runs autonomously and cooperates between AAs and resources through a social network system. Each of the AA can express different objectives and optimization criteria based on policy specification. Second, the decisions taken by the autonomic adapters are based on a rich and flexible set of high-level ECA rule-based policies represented in XML that are unavailable in existing context-aware autonomic solutions. Finally, each AA has a local rule inference engine, which triggers own AA intervention and communicates with the global rule inference engine, where a global rule inference engine combines all AAs output actions for final decision making by constantly communicating with shared knowledge source.

As a further motivation behind this research, the healthcare domain has been considered as a case study to illustrate social-based collaborative work that needs context-aware autonomic management. We demonstrated that AAs can offer continuous monitoring of a patient's health condition, the collection of patient data, and the ability to send action to the GRiST care network such as triggering critical alarms or notification when a patient's critical conditions detected so that doctors or carers can instantly react to them. We explored that social networking enabled collaborative systems are ideal for the future of e-Health systems to enable the delivery of healthcare outside clinical sites at a reduced cost and in conjunction with self-managed care network.

Similar to any other autonomic system framework, it is also important to evaluate and compare the capabilities of our proposed GASCF and AA architecture against the existing solutions. However, testing dynamically adaptive systems is extremely challenging due to the fact that the structure and behaviour of the system may change during its execution. There has not been any comprehensive work addressing evaluation criteria or metrics for the self-adaptive system. However, the availability of appropriate case studies can help in evaluating and comparing distinct adaptation solutions, at least in relation to each adaptation process, which has done in case of our proposed approach.

One of the paramount challenges that autonomic system research is currently focusing is related to describing the appropriate and comprehensive policy model and specification language. The limitation of this work is that the AA policies are configured in a standard XML data model, which is not appropriate for the efficient, flexible representation and validation of the performance of the system in many other contexts. In order to test and evaluate the correct system response according to the defined policy, we need a comprehensive policy model and specification language, and then testing it in real-world problem domains.

5.7 Conclusion

This chapter has provided powerful motivation for pursuing this programme of research to increase the applicability of autonomic systems. It has presented GASCF as a promising approach for developing a socially context-aware collaborative application. We introduced the autonomic adapter (AA) architecture, which is an intelligent agent able to monitor the status of physical resources and collaborative artifacts and suggests actions based on policies defined by the domain experts. We argue that AAs can transform traditional social-collaborative networks into an intelligent social context-aware collaborative system, which is capable of autonomous monitoring of cyber-physical and social resources. It can also analyse the system states and make an appropriate decision without human intervention to serve its purpose. The GRiST case study demonstrates the potential applicability for our generic approach for the implementation of a socially context-aware system.

In Chapter 6, we discuss how to design a comprehensive policy model and specification language and how to exploit it as shared knowledge within the rule inference engines to control AA's behaviour, and on the addition of more configurable AA application program interfaces in order to support domain-specific needs.

CHAPTER 6

Policy Model and Specification Language for AA

The policy-based approach is suitable for governing the behaviour of autonomic system operations and adjusting the services accordingly [199]. To perform policy-based self-management task, autonomic adapters (or agents) must rely on its knowledge base and execute it to fulfil the operational needs. However, implementing policy-based adaptation in the autonomic system is a complex and challenging task, which requires to design appropriate policy model and its representation and execution mechanism within autonomic computing knowledge source [36].

In previous Chapter 5, we proposed a generic autonomic framework, called GASCF and Autonomic Adapter (AA) architecture with simple XML-based policy representation strategy, introduced in Chapter 5, Section (5.2) and (5.3). In this chapter, we presents a more comprehensive and flexible Event-Condition-Action (ECA) rule-based policy model and specification language to overcome the limitation of the existing autonomic computing knowledge representation and execution approach.

This chapter is organised as follows: Section (6.1) presents a general introduction to the problem. Section (6.2) discusses the policy model and specification language with a conceptual model of ECA rule-based policy interaction and an Entity Relationship Diagram (ERD). This section also defines a policy specification syntax in XML and related Policy Schema Definition (PSD) language. Section (6.3) discusses eGRiST case study. Section (6.4) discusses the prototype implementation of the system. Section (6.5) discusses the AA Engine implementation with Java programming language. Section (6.6) discusses the testing and evaluation of the proposed system with appropriate input and output action management drawn from the eGRiST case study. Finally, Section (6.7) concludes the outcome of the work presented in this chapter.

6.1 Introduction

During the last two decades, policy-based systems have gained considerable attention within both computer science research community and IT industry. A policy describes principles or strategies for a plan of action designed to achieve a particular set of goals [150]. The main goal of the policy-based system is to perform management tasks and controls the actions of a complex system at a high-level of abstraction. The policy-based management allows administrators to modify system behaviour without changing the source code, thereby reduce the amount of work and time to be spent by human operators for management and maintenance of the system [198]. Initially, governing system behaviour using policies increasingly is used as a means of implementing flexible and adaptive systems for management of Internet services, networks and security systems [220]. Previous research into the policy-based system demonstrated that it is a suitable approach for network administration [244], network-wide configuration management [198], a self-adaptive scheme in pervasive computing [117], policy-based data integration for e-Health monitoring [78], policy-based governance of complex distributed systems [36], and many others.

Leveraging policy-based techniques to achieve self-managed control is an important way to manage the next generation of autonomic system and services [13]. Recently, there have been several pieces of research focusing on the policy-based autonomic computing system, which reveal how the collaboration or cooperation among autonomic managers at different levels can be done based on the active policies [168]. A policy defines as knowledge source for autonomous agents to decide why, when, how and where to perform the adaptation operations [202]. However, in a legacy self-managed system, adaptation strategy is done in an ad-hoc manner, which is based on predicting future circumstances and embedding the adaptation decisions in the program code in a limited variety of contexts [117]. In contrast, the algorithmic approach based on pre-defined rule sets often fails where the environment is uncertain due to the unforeseen cases, unmanageable complexity, emerging behaviour and too complicated to be predicted [87]. Thus, the policy-based approach is arguably the most flexibility and general applicability among the currently popular autonomic system approaches [12].

Within autonomic computing, low-level decision making is often designed based on the well-known Event-Condition-Action (ECA) rule-based policy model [22]. Considerable effort has been devoted to developing expressive representations and sophisticated management systems for specifying, managing, analysing, and enforcing policies. Event-Condition-Action (ECA) rules are a widely used approach for the high-level specification of controllers in autonomic computing systems [43]. ECA rules define how to perform actions in response to events if specified conditions meet [22]. However, the common use of fixed and hand-coded policies in AC knowledge component is inadequate and inflexible, when the system is constantly changing and exhibits varying or uncertain information, which makes the system management decision very difficult and inappropriate [26, 245]. Instead, the policy should be specified as externally-imposed rules whose enforcement is for a predictable system behaviour [36] and to control the adaptation behaviour at run-time in a more flexible way, outside the code itself.

Several works proposed policy specification language for domain-specific autonomic system, which are not appropriate and reconfigurable for many cases for managing cyber-physical resources in a social context. As proposed AA architecture in Chapter 5.3, we explored the true potential of autonomic computing approach within socio-cyber-physical collaborative system. However, we identified several key limitations (i.e., inflexibility and inconsistency) for the modelling and representation of autonomic computing policy. Therefore, one of the research challenge is how to define a more flexible policy model and specification language and how the policies should be executed by the AAs. Currently, there is no in-depth discussion on fulfilling the gap of appropriate policy specification and execution model for the socio-cyber-physical collaboration and autonomous decision process.

To overcome these limitation, this chapter proposes a more comprehensive and flexible policy model and specification language for AA, modelled in XML syntax that can be imposed into autonomic computing knowledge. In this effort, first, we proposed an ECA rule-based policy execution framework in the social cyber-physical environment. Afterwards, we describe an Entity Relationship Diagram (ERD) to illustrate the interrelationship between AA, managed resources and policy with their associated attributes or properties. Then, we define the Policy Schema Definition (PSD) language and Policy Specification Script (PSS) in XML representation format, a formal way to specify the structure of the policy. The PSS is an XML instance document that describes the domain-specific policy configuration in terms of PSD syntax and semantics. XML has many advantages to represent the policy for its straightforward extensibility [36] and machine readability. This policy can be deployed as an external script in the knowledge repository and translated as rules by the global rule inference engine, which are shared by the AA's intelligent manager components. Finally, we implement a prototype Java console-based application of an autonomic computing engine to test the feasibility of the proposed AA functions in terms of input and output parameter. We demonstrate and validate proposed system by applying it to the eGRiST[1] mental-health assessment system. We collected the experimental datasets and configured policies to manage three AAs operation based on the datasets. We input parameter from the case domain and observed the output actions triggered by the AA according to the rules specified in the policy.

6.2 Policy Model And Specification Language

The AA knowledge is expressed in the terms of ECA rule-based policy model and specification language that define system behaviour set by a system administrator. First, we describe

[1]https://www.egrist.org/home

the higher-level conceptual model of the ECA rule execution within the AA and adaption to the IT system. We then discuss an Entity Relationship Diagram (ERD) to illustrate the interrelationship between AA's components, managed resources and policy rules with their associated attributes or properties. Afterwards, we define the Policy Schema Definition (PSD) language, along with rules and constraints for policy content and semantics, a formal way to define policy in XML representation format. The PSD can be also be used to implement and validate the structure and consistency of the Policy Specification Script (PSS). The PSS is an XML instance document that describes the domain-specific policy configuration in terms of PSD syntax and semantics. This PSS can be deployed as an external script in the knowledge repository and translates as rules by the global rule inference engine, which are shared by the AA's intelligent manager components.

Our proposed PSD language adheres to be most flexible and domain-independent, which will allow a system administrator to define policies in any IT environment that incorporate our GASCF with AA model. The policy will regulate the AAs behaviour without modifying the code itself and disrupting the normal system operation to the target domain.

6.2.1 Conceptual Model of ECA Policy Interaction

The conceptual model illustrates how AA uses ECA rules from the policy knowledge in the application domain (i.e., managed resources or environment) in order to control some tasks at runtime. As mentioned earlier in this chapter, the fundamental construct of our proposed AA policy model is based on the ECA rule that defines when a specific event occurs, if conditions are true, then actions would be triggered. Figure 6.1 shows the conceptual model of ECA rule-based policy execution inside an AA for managing resources in an IT system.

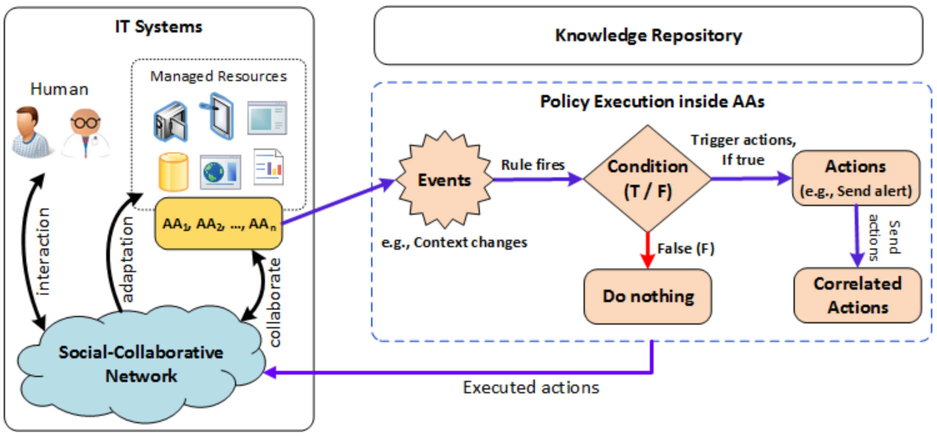


Fig. 6.1 Conceptual Model of ECA Rule Execution

As shown in Figure 6.1, the ECA rules in a policy definition consist of three main parts, namely, event, condition, and action. The *event* part dictates one or more possible states of a resource that can be occurred at regular interval. When any of the required states occurred in the managed resource at a specific point of time, an event is considered to be triggered and the rules are activated. The *rules* dictates how the evaluation of one or more condition can impact on firing actions in order to perform adaptation to the system. The condition part specifies some threshold parameters, which is evaluated against input event variables from the source environment to determines the actions to be triggered if the conditions are satisfied. The *action* part describes the changes or adaptation to be made to the target resources or environment that influences the overall system self-management. However, before the execution of the action, it is sent to the Global RIE, which correlate all actions and send back to the AA, which is then executed accordingly.

6.2.2 Entity Relationship Diagram of AA Policy Model

In general, an Entity Relationship Diagram (ERD) describes explicit logical representation of a collection of *Entity* $(E_1, E_2, ..., E_n)$, a set of *Attributes* $(attr_1, attr_2, ..., attr_n)$ of these entities and a set of *Relationship* $(R_1, R_2, ..., R_n)$ between these entities. An entity can be a person, artifacts, resources, etc. that are relevant to a given context-aware IT system, and the relationship describes how these entities interact, which are represented by an association (connector) in the ERD. The cardinality constraints define the possible number of occurrence between entities and typically are one of the following forms: one-to-one (1..1), one-to-many (1..n), many-to-many (m..n).

As depicted in Figure 6.2, in the AA policy ERD, the core entities are *AA*, *Monitor*, *LocalRIE*, *Executor*, *ManagedResource*, *Sensor*, *SocialNetwork*, *GlobalRIE*, *SharedKnowledge* and *PolicyRules.*

In this ERD, the core entities are *AA*, *Monitor*, *LocalRIE*, *Executor*, *ManagedResource*, *Sensor*, *SocialNetwork*, *GlobalRIE*, *SharedKnowledge* and *PolicyRules.* The entity *AA* has one or more *Sensor* and one particular *Sensor* is connected to one specific *AA* only. It is represented as one-to-many relationships. A particular AA associated with one *Monitor*, one *LocalRIE* and one *Executor*, which is represented as one-to-one relationship. An *AA* can manage one or more *ManagedResource* and each of the *ManagedResource* may be managed by many AA, which is represented as many-to-many relationships. A *Sensor* can sense events from one or many *ManagedResource*, represented as one-to-many relationships. On the other hand, a *Sensor* can sense events from at least one *SocialCollaboraiveNetwork* environment, represented as one-to-one relationship. One or more sensors notify the events to a *Monitor* connected with a particular *AA* and the relationship is represented as many-to-one. The association between a *Monitor* and a *LocalRIE* is one-to-one only. One or many AA's *LocalRIE* can communicate with a *GlobalRIE* by sending triggered action once at a time, which is represented as many-to-one relationship. However, a *GlobalRIE* needs to communicate with one or more AA's *Executor* at the same time to confirm the actions to be executed, and the relationship is represented as one-to-many. One or many *Executors* may

Fig. 6.2 AA Policy Specification Model

send actions to a *SocialCollaboraiveNetwork*, thus the relationship is many-to-one. At the same time, a *SocialCollaboraiveNetwork* can send adaption to one or many *ManagedResource*, which is expressed as one-to-many relationships. One or many *LocalRIEs* can access a *SharedKnowledge*, which is represented as many-to-one relationship. A *GlobalRIE* needs to access a *SharedKnowledge*, which is represented as one-to-one relationship. A *SharedKnowledge* may contain many *PolicyRules* that can be used by the AA's components, and the relationship is represented as one-to-many.

6.2.3 Policy Schema Definition (PSD) Language

The Policy Schema Definition (PSD) language is a formal way to specify the structure and syntax of the XML representation of the AA policy, along with rules and constraints for data content and semantics. The PSD language is developed based on XML Schema

Definition (XSD) artifacts [2] (e.g., element, complex types, simple type, sequence, etc.) for the mapping the policy ERD into the XML declaration syntax. The PSD consists of a set of markup declarations of the elements and sub-elements, attributes, cardinality, the datatype and values that are essential for creating XML instance documents for domain-specific policy configuration. Due to the limited space, the partial declaration of PSD language is discussed below and the complete and workable PSD is available at Github (https://github.com/nhussain920/thesis_source_code), which can be used to configure and validate any policy instance document.

The declaration of PSD language is discussed below. The complete and workable PSD is presented in Appendix B that can be used to configure and validate any policy instance without further modification of the schema.

Schema Element: Every PSD document starts with a single root *<xs:schema/>* element that belongs to the *XML Schema namespace.* The value *"http://www.aapolicyschema.com/policy.xsd"* of the *targetNamespace* attribute is simply a unique identifier where the policy schema is stored. The definition of *<schema/>* element is as follows:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.aapolicyschema.com/policy.xsd"
elementFormDefault="qualified">
  ...
</xs:schema>
```

PolicySet Element: In the following code fragment, for a given a set of policy *(PS)*, it is defined as a tuple *PS = (AAP,GP)*, where *AAP* is a finite set of AA policy, each associates with an individual autonomic adapter. *GP* is a single set of global policy, which collates all the triggered action by AAs for final intervention decision. In PSD declaration, the element *<PolicySet>* comprised of a sequence of child elements *<AAPolicy>* and *<GlobalPolicy>*, which is defined as follows:

```
<xs:element name="PolicySet">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="AAPolicy" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="GlobalPolicy"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

AAPolicy Element: For an *AAPolicy* element *(AAP)*, it is defined as a tuple *AAP = (AA, {MRl}, SN)*, where *AA* represents the autonomic adapters, *{MRl}* is the managed resource list that is a container for all declared managed resources, and *{SN}* is the social

[2] https://www.w3.org/TR/xmlschema11-1/

network that contains social entity elements. In PSD definition, the element *<AAPolicy>* comprised of a sequence of child elements *<ManagedResourceList>*, *<SocialNetwork>*, and *<AA>*. The *<AAPolicy>* element always has the required attributes *'policyID'* to define an unique id for a policy. The *'policyType'* and *'policyDescription'* are the optional attribute, and *'ref_aa'* attribute is always required that is used to define a reference value of the associated AA. The definition of *<AAPolicy>* element in PSD defined as follows:

```
<xs:element name="AAPolicy">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ManagedResourceList" minOccurs="0"/>
      <xs:element ref="SocialNetwork" minOccurs="0"/>
      <xs:element ref="AA"/>
    </xs:sequence>
    <xs:attribute type="xs:string" name="policyID" use="required"/>
    <xs:attribute type="xs:string" name="policyType" use="optional"/>
    <xs:attribute type="xs:string" name="policyDescription" use="optional"/>
    <xs:attribute type="xs:string" name="ref_aa" use="required"/>
  </xs:complexType>
  </xs:element>
```

ManagedResourceList Element: For a *ManagedResourceList* element *(MRl)*, it may contain a finite set of *ManagedResource* elements $MR = \{MR_1, MR_2, ..., MR_n\}$, which represent one or more resource that can trigger events. In PSD declaration, the element *<ManagedResourceList>* can have a sequence of child element *<ManagedResource>*, which is defined as follows:

```
<xs:element name="ManagedResourceList">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ManagedResource" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

ManagedResource Element: In a *ManagedResource* element *(MR)*, there may be a finite set of *TriggerEvent* element that represents possible events that may be triggered by resources. In PSD declaration, the element *<ManagedResource>* can have a sequence of child element *<TriggerEvent>* with a mandatory attribute *'resourceID'* to define an unique id for a managed resource. The *'resourceType'* is an optional attribute. The required attribute *'ref_sensor'* is used to define a reference value of the attached sensor, which is defined as follows:

```
<xs:element name="ManagedResource">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="TriggerEvent"/>
    </xs:sequence>
    <xs:attribute type="xs:string" name="resourceID" use="required"/>
    <xs:attribute type="xs:string" name="resourceType" use="required"/>
    <xs:attribute type="xs:string" name="ref_sensor" use="required"/>
  </xs:complexType>
</xs:element>
```

Event Element: For an *Event* element *(E)*, it has required attributes *'inputEvent'* to define an event parameter (i.e., event name or id), and *'op'* to define any arithmetic symbol (i.e., '==', '!=', '!', '>=', '<=') for the parameter.

```
<xs:element name="Event">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute type="xs:string" name="inputEvent" use="required"/>
        <xs:attribute type="xs:string" name="op" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

SocialNetwork Element: For a given *SocialNetwork* element *(SN)*, it is defined to represent a list of network resource or node elements. The element *<SocialNetwork>* can have a sequence of child element *<NetworkResourceList>* with required attributes *'networkID'* to define a unique id for the network and optional attribute *'networkName'* to specify a network name.

```
<xs:element name="SocialNetwork">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="NetworkResourceList"/>
    </xs:sequence>
    <xs:attribute type="xs:string" name="networkID" use="required"/>
    <xs:attribute type="xs:string" name="networkName" use="required"/>
  </xs:complexType>
</xs:element>
```

AA Element: For an *AA* element, it is defined as a tuple, AA = *({Sl}, M, lRIE, EX)*, where *{Sl}* represents the sensor list element that is a container for all sensor associates with a particular AA, *M* represents the monitor of this AA, *lRIE* represents the local rule inference engine, and *EX* represents the executor of the AA. In PSD declaration, the element *<AA>* is comprised of a sequence of child elements *<SensorList>*, *<Monitor>*, *<LocalRIE>*, and *<Executor>*. The *<AA>* element always has required attribute *'adapterID'* to define an unique id for an AA and an optional attribute *'adapterType'* to define the type of adapter.

```
<xs:element name="AA">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element ref="SensorList" minOccurs="0"/>
```

```
        <xs:element ref="Monitor" minOccurs="0"/>
        <xs:element ref="LocalRIE" minOccurs="0"/>
        <xs:element ref="Executor" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute type="xs:string" name="adapterID" use="required"/>
      <xs:attribute type="xs:string" name="adapterType" use="optional"/>
    </xs:complexType>
  </xs:element>
```

SensorList Element: For a given *SensorList* element *(Sl)* there may be a finite set of *Sensor* elements, which represents one or more sensor that are connected with a managed resource or network node to observe events. In PSD, the element *<SensorList>* have a sequence of child elements *<Sensor>* and is declared as follows:

```
<xs:element name="SensorList">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Sensor" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Sensor Element: For a *Sensor* element *(S)* it has required attribute *'sensorID'* to define an unique id for a sensor and an optional *'sensorType'* attribute to define the type of sensor.

```
<xs:element name="Sensor">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute type="xs:string" name="sensorID" use="required"/>
        <xs:attribute type="xs:string" name="sensorType" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Monitor Element: For a *Monitor* element *(M)*, there may be a finite set of *GetSensorData* elements to represent the data receiving from sensors. It may also have one or more *<Condition>* element to define some conditional statements to evaluate the threshold parameters. The *<Monitor>* element has a sequence of child elements *<GetSensorData>* and *<Condition>* element. The required attributes *'monitorID'* to define an unique id for a monitor element.

```
<xs:element name="Monitor">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="GetSensorData" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="Condition"/>
    </xs:sequence>
   <xs:attribute type="xs:string" name="monitorID" use="required"/>
  </xs:complexType>
</xs:element>
```

LocalRIE Element: For a *LocalRIE* element *(lRIE)*, it is a tuple *lRIE = (GMD, {C}, TA)*, where *GMD* represents data receiving from AA's monitor, *{C}* represents a list of conditional expression associates with received monitor data, and *TA* represents the action to be triggered if conditions are met. In PSD declaration, the *<LocalRIE>* element has a sequence of child elements *<GetMonitorData>*, *<Condition>* and *<TriggerAction>*. It has a required attribute *'rieID'* to define an unique id for a local RIE.

```
 <xs:element name="LocalRIE">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="GetMonitorData"/>
      <xs:element ref="Condition"/>
      <xs:element ref="TriggerAction"/>
    </xs:sequence>
   <xs:attribute type="xs:string" name="rieID" use="required"/>
  </xs:complexType>
</xs:element>
```

TriggerAction Element For a *TriggerAction* element *TA* there may be a finite set of *While* elements, which represent some condition declaration. In PSD declaration, the element *<TriggerAction>* has a sequence of child elements *<While>* with required attributes *'ref_condition'* to define a reference value of the condition element defined in *<LocalRIE>* element.

```
<xs:element name="TriggerAction">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="While" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
   <xs:attribute type="xs:string" name="ref_condition" use="required"/>
  </xs:complexType>
</xs:element>
```

ActionTemplate: For an *ActionTemplate* element *AT* it defines the action details that can be executed. It has a required attribute *'actionID'* to define an unique id for an action. The optional attribute *'actionType'* is used to define a relevant action name, and *'category'* attribute is to define the action category if necessary. The required attribute *'target_resource'* may be used to define some reference value such as resource id, name or location where the adaptation or changes may occur. It is declared in PSD as follows:

```
<xs:element name="ActionTemplate">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
       <xs:attribute type="xs:string" name="actionID" use="required"/>
```

```
        <xs:attribute type="xs:string" name="actionType" use="optional"/>
        <xs:attribute type="xs:string" name="category" use="optional"/>
        <xs:attribute type="xs:string" name="target_resource" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Executor Element: For an *Executor* element *EX* there may be one or more *GetNotification* element, which are used to express notification details received from *GlobalRIE* element. In PSD declaration, the element <*Executor*> has a sequence of child element <*GetNotification*> with required attributes *'executorID'* to define an unique id of the <*Executor*> element associated with an AA and is declared as follows:

```
<xs:element name="Executor">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="GetNotification"/>
    </xs:sequence>
    <xs:attribute type="xs:string" name="executorID"/>
  </xs:complexType>
</xs:element>
```

Execute Element: For an *Execute* element *EX* there may be one or more *Function* elements that express the function definition to be called for the execution. In PSD declaration, the element <*Execute*> have a sequence of child elements <*Function*> with required attribute *'action'* that is used to define conditional statement as attribute value to define what action to be executed after getting command form *GlobalRIE*. The *'ref_command'* attribute to define a reference value of the associated command id of *GlobalRIE*:

```
<xs:element name="Execute">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Function"/>
    </xs:sequence>
   <xs:attribute type="xs:string" name="action" use="required"/>
  <xs:attribute type="xs:string" name="ref_command" use="required"/>
 </xs:complexType>
</xs:element>
```

Function Element: For a *Function* element *F* there may be one or more *Parameter* elements that represent a set of parameters to be defined for the function. In PSD declaration, the element <*Function*> have a sequence of child element <*Parameter*> with required attributes *'functionID'* to define an unique id for a function. The *'functionName'* is used to define the method signature and declared is as follows:

```
<xs:element name="Function">
  <xs:complexType>
```

```
      <xs:sequence>
        <xs:element ref="Parameter" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute type="xs:string" name="functionID" use="required"/>
     <xs:attribute type="xs:string" name="functionName" use="required"/>
    </xs:complexType>
  </xs:element>
```

GlobalPolicy Element: For a *GlobalPolicy* element *GP* there can be one specific *GlobalRIE* element to represent the global policy declaration. In PSD, the element *<GlobalPolicy>* comprised of a sequence of child element *<GlobalRIE>*, which is defined as follows:

```
  <xs:element name="GlobalPolicy">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="GlobalRIE"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

GlobalRIE Element: For a *GlobalRIE* element *gRIE*, it is a tuple *GlobalRIE = ({AL}, C, {D})*, where *{AL}* represents all input actions coming from different AAs, *C* use to specify some conditional statement and *{D}* represents a finite set of *Decision* element to define final intervention. In PSD declaration, the element *<GlobalRIE>* have a sequence of child elements *<ActionList>*, *<Condition>*, and *<Decision>* with required attribute *'rieID'* to define an unique id for a *<GlobalRIE>* element, which is declared as follows:

```
  <xs:element name="GlobalRIE">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ActionList"/>
        <xs:element ref="Condition"/>
        <xs:element ref="Decision" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
     <xs:attribute type="xs:string" name="rieID" use="required"/>
    </xs:complexType>
  </xs:element>
```

ActionList Element: For a *ActionList* element *Al* there may be a finite set of *GetAction* element that represents a set of input actions coming form AA. In PSD declaration, the element *<ActionList>* have a sequence of child elements *<GetAction>* and is declared as follows:

```
  <xs:element name="ActionList">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="GetAction" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

Decision Element: For a *Decision* element *DC*, there may be one or more *Command* elements that represent a set of intervention command to be distribute to AA's executor according to the true conditional expression defined in *<GlobalRIE>* element. In PSD declaration, the *<Decision>* element has a sequence of child elements *<Command>* with required attribute *'ref_condition'* that references to the condition element. The *'ref_expression'* attribute to define reference value to its conditional expression element.

```
<xs:element name="Decision">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Command" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
   <xs:attribute type="xs:string" name="ref_condition" use="required"/>
   <xs:attribute type="xs:string" name="ref_expression" use="required"/>
  </xs:complexType>
</xs:element>
```

Command Element: For a *Command* element *CMD* the required attribute *'commandID'* use to assign unique id for each command to be sent to the executor. The *'message'* attribute is used to assign a string literal as a notification message. The *'target_adapter'* attribute to specify the reference value of the target AA that should receive the command from global RIE. It is declared in PSD as follows:

```
<xs:element name="Command">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute type="xs:string" name="commandID" use="required"/>
        <xs:attribute type="xs:string" name="message" use="required"/>
        <xs:attribute type="xs:string" name="target_adapter" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Condition Element: For a *Condition* element *C* it can be defined under AA's different functional elements, including *Monitor*, *LocalRIE*, and *GlobalRIE* elements. It is composed of a finite set of *Expression* elements that represents condition statements, whether composite, regular or boolean types by using any of the *and*, *or*, *not* elements. In PSD declaration, the *<Condition>* element is composed of a sequence of child element *<Expression>* with a required attribute *'conditionID'* to define an unique id for each specified condition.

```
<xs:element name="Condition">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Expression" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
  <xs:attribute type="xs:string" name="conditionID" use="required"/>
 </xs:complexType>
</xs:element>
```

Expression Element: It encapsulates a set of *If* statements with boolean operator *and*, *or*, *not* to negate the values of an expression within a filter pattern. For an *Expression* element *(Expr)*, it can define one or more *If* elements with or without boolean *and*, *or*, *not* element to encapsulate conditional statements. In PSD declaration, the *<Expression>* element has a sequence of *<If>*, *<and>*, *<or>*, *<not>* element. The required attributes *'exprID'* to define an unique id for an expression but optional *'ref_event'* attribute that may or may not be necessay to define in every expression element. However, it can be use in *Monitor* element to define a reference value from an input event. In PSD, it is declared as follows:

```
<xs:element name="Expression">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="If" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="and" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="or" minOccurs="0"/>
      <xs:element ref="not" minOccurs="0"/>
    </xs:sequence>
   <xs:attribute type="xs:string" name="exprID" use="required"/>
  <xs:attribute type="xs:string" name="ref_event" use="optional"/>
 </xs:complexType>
</xs:element>
```

If Element: For a *If* element, there are mixed attributes, some of them are required and some are optional. The attributes *'varName'* to define a variable, *'op'* attribute to define the arithmetic operators (i.e., less than (&lt;) greater than (&gt;), equal (==), not equal (!=), etc.), and *'datavalue'* attribute to assign a value for a variable. The attribute *'ref_context'* is only required to define in conditional statement within *LocalRIE* element but optional for other elements. The *'ref_input_action'* attribute is only required to define for the conditional statement declaration in the *GlobalRIE* element and optional for other elements.

```
<xs:element name="If">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute type="xs:string" name="varName" use="required"/>
        <xs:attribute type="xs:string" name="op" use="required"/>
        <xs:attribute type="xs:string" name="datavalue" use="required"/>
        <xs:attribute type="xs:string" name="ref_context" use="optional"/>
        <xs:attribute type="xs:string" name="ref_input_action" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

In the above PSD declaration, we do not list all the elements due to the limited space. However, we listed only some important elements for illustration purposes. The complete PSD declaration can be found in Appendix B.

6.2.4 Policy Specification Script (PSS) Syntax

This section describes the XML representation of the Policy Specification Script (PSS) syntax for AA, which is adheres to the PSD. We use the Backus-Naur Form (BNF) [3] notation for encoding XML grammar for the PSL syntax, which is defined as follows:

```
<PolicySet>::= Element({<AAPolicy>}, <GlobalPolicy>);
<AAPolicy>::= Element({<AA>, <ManagedResourceList>}, <SocialNetwork>), Attribute(policyID,
policyDescription], ref_aa);
<ManagedResourceList>::= Element({<ManagedResource>});
<ManagedResource>::= Element({<TriggerEvent>}), Attribute(resourceID, [resourceType], ref_s
<TriggerEvent>::=  Element({<and>}, {<or>}, {<not>}, {<Event>});
<Event>::= Attribute(<inputEvent>, <op>);
<SocialNetwork>::= Element(<NetworkResourceList>), Attribute(<networkID>, <networkName>);
<NetworkResourceList>::= Element({<NetworkResource>};
<NetworkResource>::= Element({<TriggerEvent>}), Attribute(<resourceID>, <resourceType>, <re
<AA>::= Element(<SensorList>, <Monitor>, <LocalRIE>, <Executor>), Attribute(<adapterID>, <a
<SensorList>::= Element({<Sensor>});
<Sensor>::= Attribute(<sensorID>, <sensorType>);
<Monitor>::= Element({<GetSensorData>}, <Condition>), Attribute(<monitorID>);
<GetSensorData>::= Attribute(<ref_sensor>, <ref_event>);
<LocalRIE>::= Element(<GetMonitorData>, <Condition>, <TriggerAction>), Attribute(<rieID>);
<GetMonitorData>::= Element(<{GetContextVariable>}), Attribute(<contextID>, <ref_expression
<GetContextVariable>::= Attribute(<contextID>, <ref_expression>);
<TriggerAction>::= Element(<{While>}), Attribute(<ref_condition>);
<While>::= Element(<ActionTemplate>), Attribute(<condition>);
<ActionTemplate>::= Attribute(<actionID>, <actionType>, <category>, <target_resource>);
<Executor>::= Element(<GetNotification>), Attribute(<executorID>);
<GetNotification>::= Element(<{Execute>}), Attribute(<message>, <ref_sender>);
<Execute>::= Element(<{Function>}), Attribute(<action>, <ref_command>);
<Function>::= Element(<{Parameter>}), Attribute(<functionID>, <functionName>);
<Parameter>::= Attribute(<paramName>, <datavalue>);
<GlobalPolicy>::= Element(<GlobalRIE>);
<GlobalRIE>::=  Element(<ActionList>, <Condition>, {<Decision>}), Attribute(<rieID>);
<ActionList>::= Element(<{GetAction>});
<GetAction>::= Element({<and>}, {<or>}, {<not>}, {<Action>}), Attribute(<ref_aa>);
<Action>::= Attribute(<inputActionID>);
<Decision>::= Element(<Command>), Attribute(<ref_condition>, <ref_expression>);
```

[3] http://cuiwww.unige.ch/db-research/Enseignement/analyseinfo/AboutBNF.html

```
<Command>::= Attribute(<commandID>, <message>, <target_adapter>);
<Condition>::= Element({<Expression>}, Attribute(<conditionID>);
<Expression>::= Element({<and>}, {<or>}, {<not>}, {<If>}), Attribute(<exprID>, [<ref_even
<If>::= Attribute(varName, op, datavalue, [ref_context, ref_input_action]);
<and>::= Element(<If>, [<Event>], [<Action>]);
<or>::= Element(<If>, [<Event>], [<Action>]);
<not>::= Element(<If>, [<Event>], [<Action>]);
```

6.3 Case Study - eGRiST Clinical Decision Support System

In this section, we discuss a case study to demonstrate the applicability of our proposed approach in a real-world domain, known as eGRiST, web-based mental-health risk and safety management decision support system [71]. This case study focuses on integrating AAs into eGRiST CDSS for managing risk assessments and judgement nodes and triggering appropriate action or intervention according to configured policy.

The eGRiST system uses the Galatean model of decision making, which mimics how experts use patient's cues and process the cues to arrive at accurate judgements of patient mental health condition [40, 38]. Basically, eGRiST is naturally composed of hierarchical tree nodes, points users to the right risk assessment questions to answer in the hierarchy, and provide expert advice based on their answers for early detection of mental health problems. The technique is based on the premise that the probability of different decision outcomes competes with each other for influence on the final decision. However, the system currently lacks triggering appropriate actions (i.e., sending an alert or notification) autonomously based on the given risk assessment answer by patients or practitioners in the eGRiST system. Thus, an important issue of the eGRiST system is how to detect and act upon risk escalators.

In essence, eGRiST actions are specific outputs or interventions of an assessment triggered by the risk escalator. The actions are defined in a template that specifies the action type (i.e., open-url), the template name, the specific action for that template (i.e., the particular URL to be open, the alert message, who should receive the alert), the categories associated with the action (i.e., a music action, a picture action, or a poem action), and the nodes associated with the action. The action template can be associated with many nodes, but the specific instantiation of the template is for a single node. It is immensely useful for identifying the patients' mental health situations, determine the risk status and eventually trigger actions autonomously.

Figure 6.3 by C.D. Buckingham [38] shows a hypothetical assignment of membership grades and relative influences to the intended concept and how these generate membership grades for a particular patient's values.

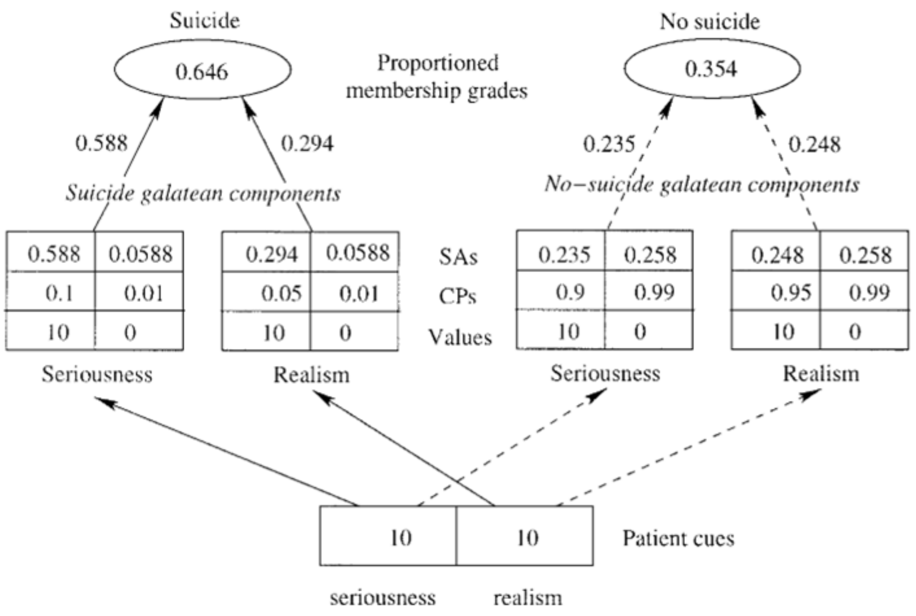


Fig. 6.3 Hypothetical example of the galatean model's classification of a patient into suicide and no-suicide categories. CPs=conditional probabilities and SAs=selective attentions [38].

6.3.1 Limitation of eGRiST Actions

Originally, eGRiST is naturally composed of hierarchical tree nodes, which can be controlled by collaborative and cooperative AAs interaction and decisions to solve the existing intervention problem.

Currently, actions are triggered by the Membership Grade (MG) of a specific node, which is generated by the provided assessment answer (cue). Meanwhile, the system handles various action types using a static database design with one-table-fits-all solutions, which is not convenient. For every new action type that is added to the system, new tables, specific to that action need to be added. At present, this means they cannot be influenced by the answers from several nodes at the same time, unless the node, they are in is a concept node. This is because concept nodes collate MGs from all of their input leaf nodes. Nevertheless, this is the only way collective answers can have an effect, and it depends on the way the MG for the concept is calculated. More generally, an MR could be a risk node, because the trigger for an event is the MG, not the specific answer.

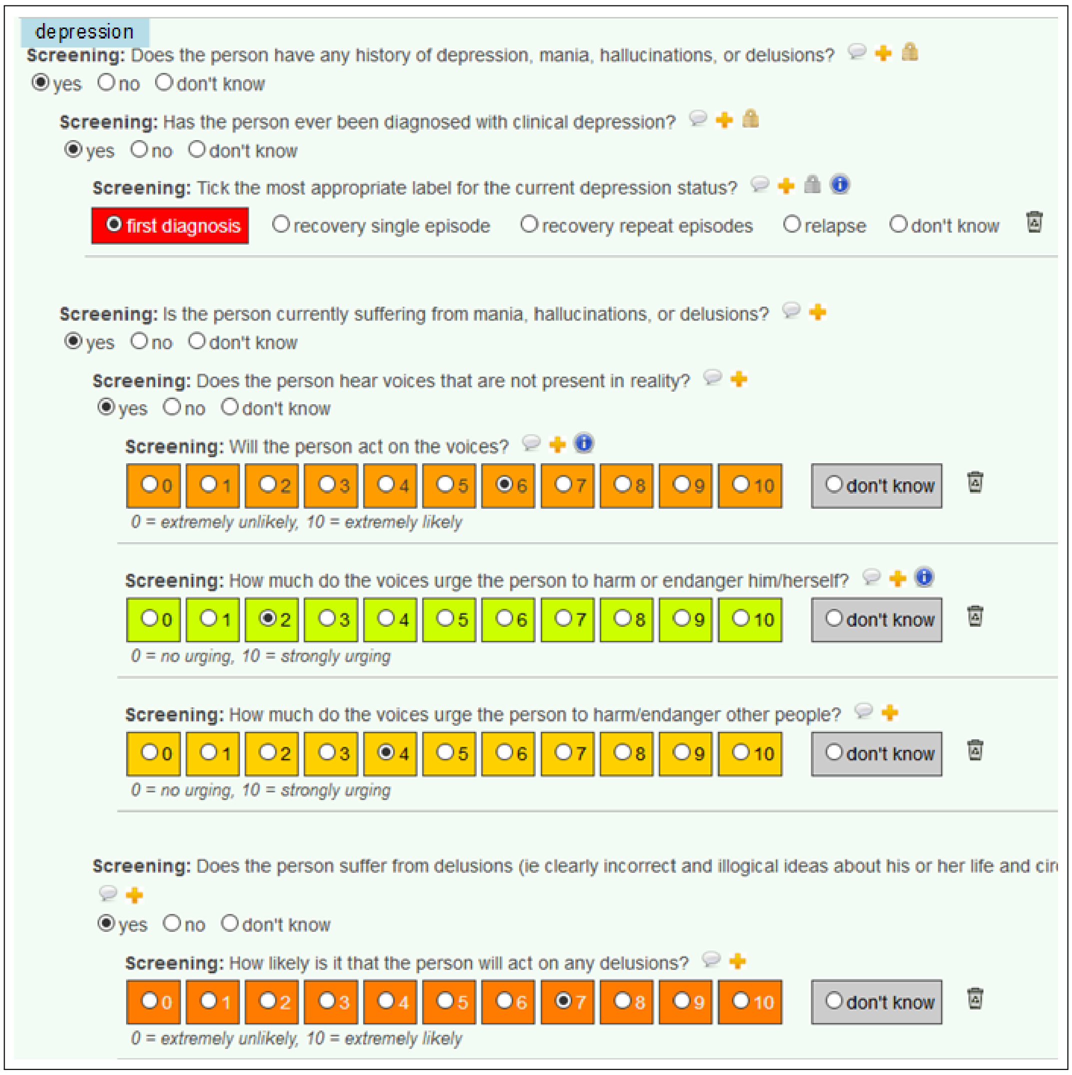
Therefore, the action may need more flexibility than being generated by the MGs, especially, if the MGs are only calculated using risk evaluation. In theory, a single intervention could be triggered by a pattern of responding across many event variables that lie the different parts of the tree. To this extent, if AAs are used for actions, each node needs the potential of "listening to" many managed resources (MRs), where an MR in this case study is taken to be an assessment answer from leaf nodes. AAs would seem to be most suited for the triggering and execution action functionality according to the externally imposed policy and based on the individual's situation or some environmental context that has been embedded within eGRiST. It is likely that interventions may need to be collated with respect to other actions in place from outside sources that can be handled using the action rules, which specify the final interventions. Hence, integrating information within the AA and also controlling actions at a higher level through the policy is the most appropriate solution. The proposed policy model can be used to configure policies integrated AAs into the eGRiST system, and AA process the data to make a collaborative intervention decision according to the defined policy.

6.3.2 eGRiST Risk Assessment Scenarios

The common practice of eGRiST system is to evaluate multiple types of mental health-related risks, including suicide, self-harm, harm to others, self-neglect, depression, vulnerability, etc., which are validated against the psychological model. To answer an assessment question, a user clicks on the radio button corresponding to the appropriate answer (cue) that is input MG variables on a scale from zero to ten, while others require yes/no/don't know answers. For example, there is a high-level filter question asking whether the patient has made any past suicide attempts. If the answer is selected as "no", this part of the tree is immediately closed off with no additional exploration. If, however, clinicians or patients say "yes', then further questions open up to carry out the assessment in this top-down fashion. Each of these selected answer options generates an MG that produces a derived MG at the concept node level. This derived MG is based on the risk judgement dependent variable (DV) and a linear regression calculation. Each cue that lies within a particular concept can have values between 0 and 10, with a value of 10 providing the maximum MG and value 0 the minimum MG. For example, if the clinician gives realism a value of 7, then the membership grade will be 0.7. Each assessment question has two additional elements: actions - allow to enter text, which describes an action that can be taken to help manage the risk issue, and comments - provide to write a comment about the understanding of that risk item. When all the information collected it needs to evaluate each risk, provide a risk judgement for each one, and create a risk formulation to produce the safety plan for the complete assessment.

Let us consider the screening questions of three concept nodes: *"depression"*, *"suicide"* and *"social world"* to demonstrate the use case scenario of eGRiST risk assessment population for older adults. These concept nodes (attached with AA) have a number of input variables comes from different leaf nodes or screening questions (consider as manage resources, attached with AAs sensors), which can trigger an event using the MG for that node and AAs sensors observe these events.

For a given *"depression"* concept, assume that the patient has been diagnosed with clinical depression previously. For the risk assessment questions, the answer selected as *clinical depression = yes*, *suffers from delusions = yes* and *act on delusions = yes.* The patient's most appropriate label for the current depression status selected as *first depression* *stage.* The MG is derived from the given answer of each cue, given as *act on the voices* $>$ *0.6*, *harm or endanger him/herself* $<$ *0.2*, *harm or endanger other people* $<$ *0.4*, and *act on any delusion* $>$ *0.7*. The Figure 6.4 shows the screenshot of the possible assessment answers given by a practitioner for the *"depression"* concept node.



depression

Screening: Does the person have any history of depression, mania, hallucinations, or delusions?

[✓ yes] no don't know

Screening: Has the person ever been diagnosed with clinical depression?

[✓ yes] no don't know

Screening: Tick the most appropriate label for the current depression status?

[✓ first diagnosis] recovery single episode recovery repeat episodes relapse don't know

Screening: Is the person currently suffering from mania, hallucinations, or delusions?

[✓ yes] no don't know

Screening: Does the person hear voices that are not present in reality?

[✓ yes] no don't know

Screening: Will the person act on the voices?

0 1 2 3 4 5 [✓ 6] 7 8 9 10 don't know

0 = extremely unlikely, 10 = extremely likely

Screening: How much do the voices urge the person to harm or endanger him/herself?

0 1 [✓ 2] 3 4 5 6 7 8 9 10 don't know

0 = no urging, 10 = strongly urging

Screening: How much do the voices urge the person to harm/endanger other people?

0 1 2 3 [✓ 4] 5 6 7 8 9 10 don't know

0 = no urging, 10 = strongly urging

Screening: Does the person suffer from delusions (ie clearly incorrect and illogical ideas about his or her life and cir

[✓ yes] no don't know

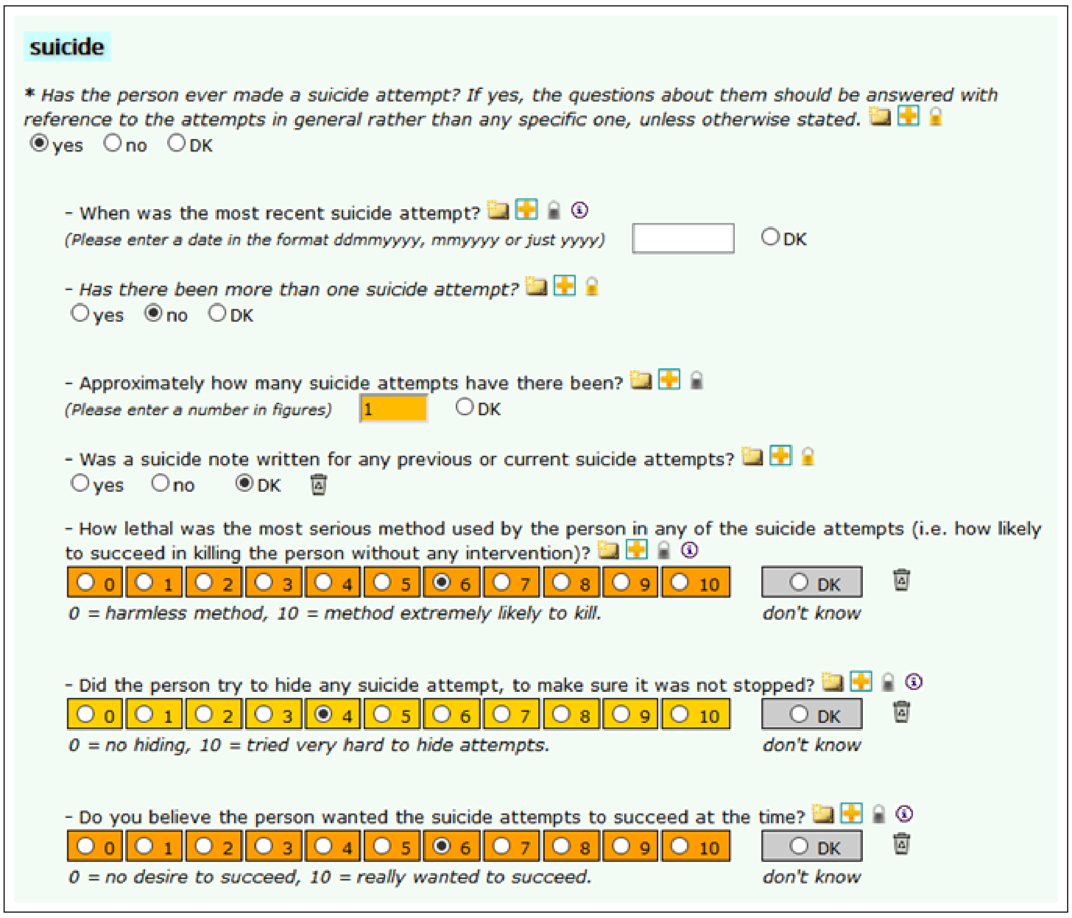
Screening: How likely is it that the person will act on any delusions?

0 1 2 3 4 5 6 [✓ 7] 8 9 10 don't know

0 = extremely unlikely, 10 = extremely likely

Fig. 6.4 Assessment Answer Given for Depression Concept

On the other hand, for the *"suicide"* concept, the assessment answer selected as *suicide attempt* = *yes*, *more than one suicide attempt* = *no*, *number of suicide attempt* = *1*, and *any suicide note written* = *don't know*. The MG is derived from the given answer of each cue, including *serious method used for suicide attempt* $>$ *0.6*, *try to hide any suicide attempt* $<$ *0.4*, and *suicide attempts to succeed* $>$ *0.6*. The Figure 6.5 shows the screenshot of the possible assessment answers given by a practitioner for the *"suicide"* concept node.



suicide

** Has the person ever made a suicide attempt? If yes, the questions about them should be answered with reference to the attempts in general rather than any specific one, unless otherwise stated.*

[✓ yes] no DK

- When was the most recent suicide attempt?

(Please enter a date in the format ddmmyyyy, mmyyyy or just yyyy) DK

- *Has there been more than one suicide attempt?*

yes [✓ no] DK

- Approximately how many suicide attempts have there been?

(Please enter a number in figures) 1 DK

- Was a suicide note written for any previous or current suicide attempts?

yes no [✓ DK]

- How lethal was the most serious method used by the person in any of the suicide attempts (i.e. how likely to succeed in killing the person without any intervention)?

0 1 2 3 4 5 [✓ 6] 7 8 9 10 DK

0 = harmless method, 10 = method extremely likely to kill. *don't know*

- Did the person try to hide any suicide attempt, to make sure it was not stopped?

0 1 2 3 [✓ 4] 5 6 7 8 9 10 DK

0 = no hiding, 10 = tried very hard to hide attempts. *don't know*

- Do you believe the person wanted the suicide attempts to succeed at the time?

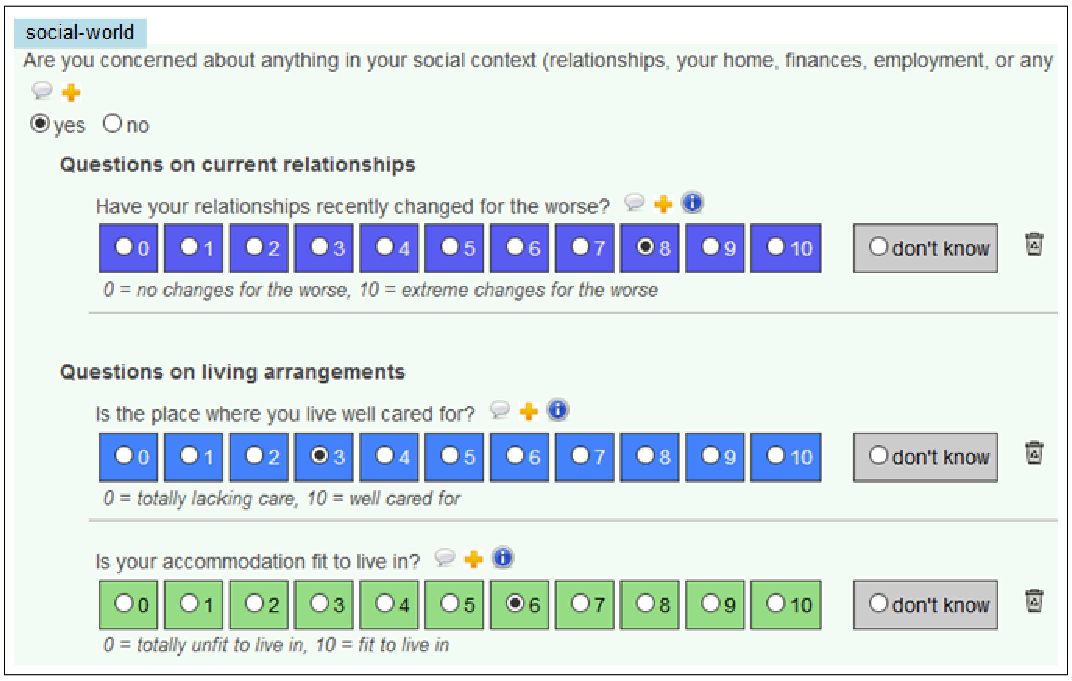
0 1 2 3 4 5 [✓ 6] 7 8 9 10 DK

0 = no desire to succeed, 10 = really wanted to succeed. *don't know*

Fig. 6.5 Assessment Answer Given for Suicide Concept

Finally, for the *"social world"* concept, the assessment answer selected as *concerned about social context* = *yes*, and the MG is derived from the given answer of each cue, including *relationships changed* > 0.8, *well cared living place* < 0.3, and *well fitted accommodation* > 0.6. The Figure 6.6 shows the screenshot of the possible assessment answers given by a patient for the *"social"* concept node.

Based on the assessment questions answered, an intervention plan depends on the input variables coming from different Managed Resources (MRs) and an MR can contribute to many interventions. The MR may have many different events (MG ranges) for each question selected, each one associated with a different action. For example, if a question at depression node asking whether the patient has a sad factor and if the answer is selected as the MG range sad > 0.5, then an action will be triggered. In this case, our AA approach can integrate risk assessments and judgements score, and then trigger an appropriate action according to the specified condition in the policy. The benefits of integrating AAs is that they can reduce the complexity of manual management of thousands of actions, possible cost, time, human errors, and accelerate faster clinical decision. Although the AA needs to link each event to


social-world

Are you concerned about anything in your social context (relationships, your home, finances, employment, or any

[✓ yes] no

Questions on current relationships

Have your relationships recently changed for the worse?

0 1 2 3 4 5 6 7 [✓ 8] 9 10 don't know

0 = no changes for the worse, 10 = extreme changes for the worse

Questions on living arrangements

Is the place where you live well cared for?

0 1 2 [✓ 3] 4 5 6 7 8 9 10 don't know

0 = totally lacking care, 10 = well cared for

Is your accommodation fit to live in?

0 1 2 3 4 5 [✓ 6] 7 8 9 10 don't know

0 = totally unfit to live in, 10 = fit to live in

Fig. 6.6 Assessment Answer Given for Social Concept

each action, the action itself is actually defined, triggered, and executed by inputs from many other AAs. An AA cannot, therefore, activate the action on its own, which can be done by collating actions across many other AAs by the global rule inference engine. However, AAs can only be managed at the local level by local RIE, both for triggering and executing the action, which counters the idea of distributed and independent AAs.

6.3.3 eGRiST Datasets

Here, we discuss the collected data files that we use to configure the policy and input for the AAs. From these data files, we synthesised the data to use these as inputs for autonomic adapters that then return identifiers for the actions that should be fired. The data files used for the inputs includes are as follows:

- answers.xml - sample answers for an individual assessment that trigger actions.
- actions-data.json - a list of actions and categories that policies need to cater to trigger only actions in specified categories for individual calls.
- questions.xml - contains the risk assessment question text and codes.

6.4 Prototype Implementation

This section presents a prototype implementation of the Autonomic Adapter architecture introduced in Chapter 5. It also discusses the implementation and configuration of the proposed policy model in a real-world eGRiST CDSS. We deploy and configure AAs policies according to the collected data from the eGRiST and scenario discussed in previous Section (6.3.2).

6.4.1 Integrating AAs into eGRiST Domain

This section discusses the integration of AAs and policy specification according to the eGRiST scenario discussed above. As shown in Table 6.1, AA1 is integrated with the depression node, and the associated sensors *S1 to S5* are responsible for observing the means of MG input variables from connected leaf nodes (managed resources or cues). AA2 is integrated with the suicide node, and connected sensors *S1 to S7* are liable for observing the means of MG input variables from connected leaf nodes. AA3 is integrated with the social world concept, and associated sensors *S1 to S3* are responsible for observing the means of MG input variables from the connected leaf nodes.

Table 6.1 eGRiST Concept Nodes with Integrated AAs

Adapter Name	**Concept Node**	**Managed Resources**	**Sensors**	**MG Score**
AA1	Depression Node	depression-status	S1	first-diagnosis
		act-on-voices	S2	0.6
		voice-urge-harm-self	S3	0.2
		voice-urge-harm-others	S4	0.4
		delusion	S5	0.7
AA2	Suicide Node	Suicide attempt	S1	yes
		More-than-one	S2	no
		total-suicide-attempt	S3	1
		suicide-note	S4	unknown
		seriousness	S5	0.6
		hide-attempts	S6	0.4
		wanted-to-succeed	S7	0.6
AA3	Social Node	worse-relationship	S1	0.8
		well-cared-living	S2	0.3
		well-fitted-accommodation	S3	0.6

6.4.2 Policy Specifications for AAs

The ECA rule pattern is used to configure the policies for AAs, integrated with eGRiST concept nodes to manage risk assessment questions and actions. To recall AA architecture in Chapter 5, sensor observes and retrieves the resource's event such as MG score for a given cue. The monitor component receives this sensor data and looks at the rules to match the sensor data with threshold parameters. The local RIE evaluates the context variables and then triggers some action, which reflects adaptation. The behaviour of the AAs individual component is controlled by the policies stored in the knowledge repository. Each policy has a set of rules containing specific conditions which, if matched, can trigger some actions. The rule specification for each AA and Global RIE in eGRiST system are as follows:

AA1 Policy Specification This section presents the ECA rules specification for the AA1's *Monitor*, *Local RIE* and *Executor* components are shown in Tables (6.2, 6.3 and 6.4), respectively.

- AA1's Monitor policy defines fourteen rules (*R1, R2, R3, ..., R14*) and each rule is associated with a managed resource (cue) as shown in Table 6.2. The rule states that on click in an assessment question (managed resource) and if the condition part matches with the event variable, then a new context variable to be set. For example, rule *R1* is triggered if a user clicks on the managed resource "depression-status" and if the option selected is "first-diagnosis" as an assessment answer, the context variable to be set to "low".

Table 6.2 AA1 Monitor Policy in ECA Rule Pattern

Policy Name	**Rule ID**	**Event**	**Condition**	**Action**
AA1 Monitor	R1	On click depression-status cue	If depression-status == "first-diagnosis"	Set new context parameter = "low"
	R2	On click depression-status cue	If depression-status == "recovery-single-episode"	Set new context parameter = "medium"
	R3	On click depression-status cue	If depression-status == "recovery-repeat-episode"	Set new context parameter = "high"
	R4	On click depression-status cue	If depression-status == "relapse"	Set new context parameter = "critical"
	R5	On click depression-status cue	If depression-status == "unknown"	Set new context parameter = "unknown"
	R6	On click act-on-voices cue	If act-on-voices >= 0.5	Set new context parameter = "likely"
	R7	On click act-on-voices cue	If act-on-voices < 0.5	Set new context parameter = "unlikely"
	R8	On click act-on-voices cue	If act-on-voices == "unknown"	Set new context parameter = "unknown"
	R9	On click voice-urge-harm-self cue	If voice-urge-harm-self >= 0.5	Set new context parameter = "likely"
	R10	On click voice-urge-harm-self cue	If voice-urge-harm-self < 0.5	Set new context parameter = "unlikely"
	R11	On click voice-urge-harm-self cue	If voice-urge-harm-self == "unknown"	Set new context parameter = "unknown"
	R12	On click delusion cue	If delusion >= 0.5	Set new context parameter = "likely"
	R13	On click delusion cue	If delusion < 0.5	Set new context parameter = "unlikely"
	R14	On click delusion cue	If delusion == "unknown"	Set new context parameter = "unknown"

- AA1's Local RIE policy defines three rules (*R1, R2 and R3*) and each rule has associated ECA specification. The rules are trigger upon receiving a notification from monitor and if a set of context variables evaluates to be true, then trigger an action. As shown in Table 6.3, if rule *R1* get triggered, it evaluates the conditions as depression-status = "low" and act-on-voices = "likely" and voice-urge-harm-self = "unlikely" and voice-urge-harm-others = "unlikely" and delusion = "likely", then trigger "open-url" action.

Table 6.3 AA1 Local RIE Policy in ECA Rule Pattern

Policy Name	Rule ID	Event	Condition	Action
	R1	On trigger rule	If depression-status == "low" and act-on-voices == "unlikely" and voice-urge-harm-self == "unlikely" and voice-urge-harm-others == "unlikely" and delusions == "unlikely"	Trigger Action, Send Email
AA1 Local RIE	R2	On trigger rule	If depression-status == "high" and act-on-voices == "unlikely" and voice-urge-harm-self == "unlikely" and voice-urge-harm-others == "unlikely" and delusions == "unlikely"	Trigger Action, Open URL
	R3	On trigger rule	If depression-status == "critical" and act-on-voices == "likely" and voice-urge-harm-self == "likely" and voice-urge-harm-others == "likely" and delusions == "likely"	Trigger Action, Dial Emergency

- AA1's Executor policy defines three rules (*R1, R2 and R3*) and each rule has associated ECA specification. The rules are triggered upon receiving a notification from Global RIE, and the condition determines what action should be executed based on the Global RIE command. As shown in Table 6.4, if rule *R1* get triggered, the condition defines if the action id is "a1", then call "sendEmail" function to be executed.

Table 6.4 AA1 Executor Policy in ECA Rule Pattern

Policy Name	Rule ID	Event	Condition	Action
	R1	On received notification	If action == "a1"	execute action, sendEmail()
AA1 Executor	R2	On received notification	If action == "a2"	execute action, openURL()
	R3	On received notification	If action == "a3"	execute action, dial999()

AA2 Policy Specification Similar to AA1 policy, the ECA rule based policy specification for AA2's *Monitor*, *LocalRIE* and *Executor* components are shown in Tables (6.5, 6.6 and 6.7), respectively.

- AA2's Monitor policy defines twenty rules (*R1, R2, R3, ..., R21*) and each rule is associated with a managed resource as shown in Table 6.5, which are triggered based on the event from assessment questions and new context variable are set if conditions are matched.

- AA2's Local RIE policy defines two rules (*R1 and R2*), which determines what actions will be triggered if a set of context variables evaluates to true as shown in Table 6.6.

Table 6.5 AA2 Monitor Policy in ECA Rule Pattern

Policy Name	Rule ID	Event	Condition	Action
AA2 Monitor	R1	On select suicide-attempt cue	If suicide-attempt == "yes"	Set context parameter = "likely"
	R2	On select cue suicide-attempt	If suicide-attempt == "no"	Set context parameter = "unlikely"
	R3	On select cue suicide-attempt	If suicide-attempt == "unknown"	Set context parameter = "unknown"
	R4	On select more-than-one	If more-than-one == "yes"	Set context parameter = "high"
	R5	On select cue more-than-one	If more-than-one == "no"	Set context parameter = "medium"
	R6	On select more-than-one	If more-than-one= "don't know"	Set context parameter = "unknown"
	R7	On select total-attempt	If total-attempt >= 3	Set context parameter = "likely"
	R8	On select cue total-attempt	If total-attempt < 3	Set context parameter = "unlikely"
	R9	On select cue total-attempt	If total-attempt == "don't know"	Set context parameter = "unknown"
	R10	On select cue suicide-note	If suicide-note == "yes"	Set context parameter = "unlikely"
	R11	On select cue suicide-note	If suicide-note == "no"	Set context parameter = "likely"
	R12	On select cue suicide-note	If suicide-note == "don't know"	Set context parameter = "unknown"
	R13	On select seriousness	If seriousness >= 0.5	Set context parameter = "likely"
	R14	On select cue seriousness	If seriousness < 0.5	Set context parameter = "harmless"
	R15	On select cue seriousness	If seriousness == "don't know"	Set context parameter = "unknown"
	R16	On select cue hide-suicide-attempt	If hide-suicide-attempt >= 0.5	Set context parameter = "hiding"
	R17	On select cue hide-suicide-attempt	If hide-suicide-attempt < 0.5	Set context parameter = "no-hiding"
	R18	On select cue hide-suicide-attempt	If hide-suicide-attempt == "don't know"	Set context parameter = "unknown"
	R19	On select cue wanted-to-succeed	If wanted-to-succeed >= 0.5	Set context parameter = "desire-succeed"
	R20	On select cue wanted-to-succeed	If wanted-to-succeed < 0.5	Set context parameter = "no-desire"
	R21	On select cue wanted-to-succeed	If wanted-to-succeed == "don't know"	Set context parameter = "unknown"

Table 6.6 AA2 Local RIE Policy in ECA Rule Pattern

Policy Name	Rule ID	Event	Condition	Action
AA2 Local RIE	R1	On trigger rule	If suicide-attempt == "likely" and more-than-one == "high" and total-attempt == "likely" and suicide-note == "unlikely" and seriousness == "likely" and hide-suicide-attempt == "hiding" and wanted-to-succeed == "desire-succeed"	Trigger Action, Urgent Support
	R2	On trigger rule	If suicide-attempt == "likely" and more-than-one == "high" and total-attempt == "likely" and suicide-note == "unlikely" and seriousness == "likely" and hide-suicide-attempt == "hiding" and wanted-to-succeed == "desire-succeed"	Trigger Action, Send SMS

- AA2's Executor policy defines two rules (*R1 and R2*), which dictates what action will be executed upon receiving a confirmation from Global RIE as shown in Table 6.7.

Table 6.7 AA2 Executor Policy in ECA Rule Pattern

Policy Name	**Rule ID**	**Event**	**Condition**	**Action**
AA2 Executor	R1	On received notification	If action == "a1"	execute action, urgentSupport()
	R2	On received notification	If action == "a2"	execute action, sendSMS()

AA3 Policy Specification Similar to AA1 and AA2 policy, the ECA rule based policy specification for AA3's *Monitor*, *LocalRIE* and *Executor* components are shown in Tables (6.8, 6.9 and 6.10), respectively.

- AA3's Monitor policy defines nine rules (*R1, R2, R3, ..., R9*) as shown in Table 6.8.

Table 6.8 AA3 Monitor Policy in ECA Rule Pattern

Policy Name	**Rule ID**	**Event**	**Condition**	**Action**
AA3 Monitor	R1	On select cue worse-relationship	If worse-relationship >= 0.5	Set context parameter = extremely-worse"
	R2	On select cue seriousness	If worse-relationship < 0.5	Set context parameter = "less-worse"
	R3	On select cue seriousness	If worse-relationship == "don't know"	Set context parameter = "unknown"
	R4	On select cue hide-suicide-attempt	If well-cared-living >= 0.5	Set context parameter = "well-cared"
	R5	On select cue hide-suicide-attempt	If well-cared-living < 0.5	Set context parameter = "less-cared"
	R6	On select cue hide-suicide-attempt	If well-cared-living == "don't know"	Set context parameter = "unknown"
	R7	On select cue wanted-to-succeed	If well-fitted-accommodation >= 0.5	Set context parameter = "well-fitted"
	R8	On select cue wanted-to-succeed	If well-fitted-accommodation < 0.5	Set context parameter = "unfit"
	R9	On select cue wanted-to-succeed	If well-fitted-accommodation == "don't know"	Set context parameter = "unknown"

- AA3's Local RIE policy defines two rules (*R1 and R2*), as shown in Table 6.9.

Table 6.9 AA3 Local RIE Policy in ECA Rule Pattern

Policy Name	**Rule ID**	**Event**	**Condition**	**Action**
AA3 Local RIE	R1	On trigger rule	If worse-relationship == "less-worse" and well-cared-living == "well-cared" and well-fitted-accommodation == "well-fitted"	Trigger Action, Send SMS
	R2	On trigger rule	If worse-relationship == extremely-worse" and well-cared-living == "less-cared" and well-fitted-accommodation == "unfit"	Trigger Action, Call Social Service

- AA3's Executor policy defines two rules (*R1 and R2*), as shown in Table 6.10.

Table 6.10 AA3 Executor Policy in ECA Rule Pattern

Policy Name	Rule ID	Event	Condition	Action
AA3 Executor	R1	On received notification	If action == "a1"	execute action, sendSMS(Patient 10)
	R2	On received notification	If action == "a2"	execute action, callSocialService()

Global RIE Policy Specification The Global RIE policy defines four rules (*R1, R2, R3 and R4*) and that rules are triggered upon receiving actions from the one or more AA's local RIE. These rules are most important to determine the final intervention by collating all input actions triggered by the AAs. The ECA rule specification for Global RIE that determine the factors that influence decision making is shown in Table 6.11.

Table 6.11 Global RIE Policy in ECA Rule Pattern

Policy Name	Rule ID	Event	Condition	Action
Global RIE	R1	On triggered action	If (aa1.action == "a1" aa2.action == "a1" aa3.action == "a1")	send confirmation command to AA1, AA2 and AA3 to execute their triggered action "a1".
	R2	On triggered action	If aa1.action == "a2" aa2.action == "a2" aa3.action == "a2"	send confirmation command to AA1, AA2 and AA3 to execute their triggered action "a2".
	R3	On triggered action	If aa1.action == "a3" aa2.action == "a1" aa3.action == "a2"	send confirmation command to AA1 to execute action "a3" and AA2 to execute action "a1", but not AA3 action "a2" .
	R4	On triggered action	If aa1.action == "a2" aa2.action == "a1" aa3.action == "a1"	send confirmation command to AA2 and AA3 to execute both action "a1", but not AA1 action "a2".

Based on the ECA rule specification for AA1, AA2 and AA3, in the following Section 6.4.2.1, we configure the XML representation of the policies that enforce PSL syntax and is validated by PSD.

6.4.2.1 XML Configuration of AAs Policies

This section presents the XML configuration of AA1 policy, which is deployed into eGRiST CDSS as shown partially in the listings 6.2, 6.1, 6.3, 6.4 and 6.5. Due to the limited space, we do not show the policy configuration for AA2, AA3 and Global policy in this section. However, the full configuration of all of the policies policy that enforces PSD and PSS syntax is shown in Appendix B. Additionally, the XML representation of the configured policies is available at the URL : https://github.com/nhussain920/thesis_source_code. All the configured policies are validated against PSD language, which is consistent, valid and available to use for testing.

```
<!-- PolicySet root element -->
<PolicySet xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.aapolicyschema.com/␣policy.xsd">
 <!-- Define events dandle by AA1 policy -->
 <AAPolicy policyID="p1" policyType="depression"
  policyDescription="carer's␣medium␣help␣intervention" ref_aa="aa1">
 <ManagedResourceList>
```

```
  <ManagedResource resourceID="mr1" resourceType="depression-status"
      ref_sensor="s1">
   <TriggerEvent>
    <or><Event inputEvent="depression-status" op="="> first-diagnosis </Event>
        </or>
    <or><Event inputEvent="depression-status" op="="> recovery-single-episode
        </Event></or>
     ...
   </TriggerEvent>
  </ManagedResource>
  <ManagedResource resourceID="mr2" resourceType="act-on-voices" ref_sensor="
      s2">
   <TriggerEvent>
    <or><Event inputEvent="act-on-voices" op="="> 0.0 </Event></or>
    <or><Event inputEvent="act-on-voices" op="="> 0.1 </Event></or>
     ...
     ...
   </TriggerEvent>
  </ManagedResource>
  ...
 ...
</PolicySet>
```

Listing 6.1 : XML Configuration for AA1's event Policy.

```
<!-- Configure AA1 sensor policy -->
<AA adapterID="aa1" adapterType="depression-concept-adapter">
 <SensorList>
  <Sensor sensorID="s1" sensorType="depression-status-sensor" />
  <Sensor sensorID="s2" sensorType="act-on-voices-sensor" />
   ...
 </SensorList>
 ...
</AA>
```

Listing 6.2 : XML Configuration for AA1's sensor Policy.

```
<!-- Configure AA1 monitor Policy -->
<Monitor monitorID="m1">
   <GetSensorData ref_sensor="s1" ref_event="depression-status" />
     ...
      <Condition conditionID="c1">
      <Expression exprID="expr1" ref_event="depression-status">
         <If varName="depression-status" op="==" threshold="first-diagnosis">
             low </If>
          ...
      </Expression>
      <Expression exprID="expr2" ref_event="act-on-voices">
         <If varName="act-on-voices" op="&gt;=" threshold="0.5"> likely </If>
          ...
      </Expression>
          ...
          ...
 </Condition>
</Monitor>
```

Listing 6.3 : XML Configuration for AA1's Monitor Policy.

```
<!-- Configure AA1 local RIE policy -->
<LocalRIE rieID="aa-rie1">
   <GetMonitorData ref_monitor="m1" ref_condition="c1">
      <GetContextVariable contextID="context1" ref_expression="expr1"/>
      <GetContextVariable contextID="context2" ref_expression="expr2"/>
       ...
   </GetMonitorData>
   <Condition conditionID="c2">
      <Expression exprID="expr1">
         <and><If varName="depression-status" op="==" datavalue="low"
             ref_context="context1"> true </If></and>
         <and><If varName="act-on-voices" op="==" datavalue="likely"
             ref_context="context2"> true </If></and>
          ...
      </Expression>
           ...
           ...
   <TriggerAction ref_condition="c2">
      <While condition="expr1␣=␣'true'">
         <ActionTemplate actionID="a1" actionType="open-url" category="none"
             target_resource="resource␣name">
             galassify-action-open-url-template
         </ActionTemplate>
      </While>
      <While condition="expr2␣=␣'true'">
         <ActionTemplate actionID="a2" actionType="send-email" category="none"
              target_resource="resource␣name">
             galassify-action-send-email-template
         </ActionTemplate>
      </While>
       ...
   </TriggerAction>
</LocalRIE>
```

Listing 6.4 : XML Configuration of AA1's Local RIE Policy.

```
<!-- Configure AA1 local RIE policy -->
<GlobalPolicy>
  <GlobalRIE rieID="globalRIE">
    <ActionList>
      <!-- Assume that send-email action triggered by AA1 -->
      <GetAction ref_aa="aa1">
         <or><Action inputActionID="aa1-action"> a1 </Action></or>
         <or><Action inputActionID="aa1-action"> a2 </Action></or>
         <or><Action inputActionID="aa1-action"> a3 </Action></or>
      </GetAction>
      <GetAction ref_aa="aa2">
     ...
      </GetAction>
   ...
    </ActionList>
  <Condition conditionID="c3">
    <Expression exprID="expr1">
      <and><If varName="action" op="==" datavalue="a1" ref_input_action="
          aa1_action"> true </If></and>
      <and><If varName="action" op="==" datavalue="a1" ref_input_action="
          aa2_action"> true </If></and>
    </Expression>
    <Expression exprID="expr2">
      ...
    </Expression>
    ...
  </Condition>
  <Decision ref_condition="c3" ref_expression="expr1">
    <Command commandID="cmd1" message="yes" target_adapter="aa1"> a1 </Command
        >
    <Command commandID="cmd2" message="yes" target_adapter="aa2"> a1 </Command
        >
   </Decision>
    ...
   </Decision>
  </GlobalRIE>
</GlobalPolicy>
```

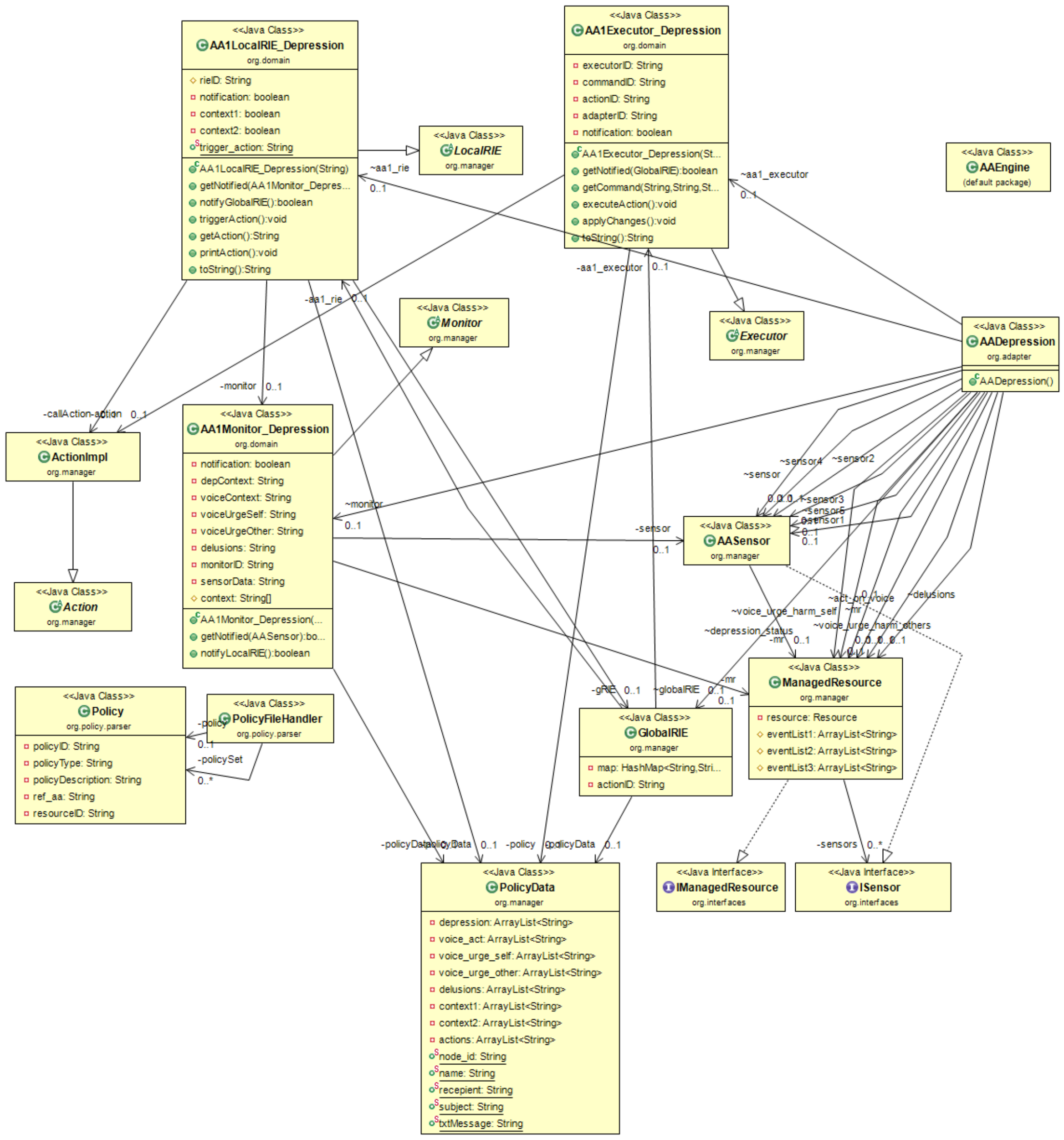
Listing 6.5 : XML Configuration of Global RIE Policy.

6.5 Implementing AA Engine With JAVA

This section presents a prototype implementation of “AA Engine” with Java console-based application. The “AA Engine” design pattern includes a set of abstract classes, interfaces, data members and methods that provide a generic solution to create AAs to attached with managed resources within the application domain. The advantage of our “AA Engine” design patterns is that the code is easily extensible, reusable and maintainable in other similar autonomic system application. The common Java abstract classes and interfaces are defined for “AA Engine” includes interfaces (i.e., ISensor, IManagedResource, INotification) and concrete classes (i.e., AASensor and ManagedResource) that implements interfaces; abstract classes (i.e., Monitor, LocalRIE, Executor) and concrete classes (i.e., AA1Monitor_Depression, AA1LocalRIE_Depression, AA1Executor_Depression) that extends the abstract classes. The Unified Modelling Language (UML) class diagram of “AA Engine” in Figure 6.7 shows the association and dependencies between the classes.

6.5.1 AA Engine Design Pattern

The following subsections and code fragment show the partial implementation of “AA Engine” design pattern with Java, whcih can be extendted to create domain specific AAs classes and interface. The full source code is available online, which can be found using the URL link: https://github.com/nhussain920/thesis_source_code.



<<Java Class>> AA1LocalRIE_Depression org.domain
rieID: String
notification: boolean
context1: boolean
context2: boolean
trigger_action: String
AA1LocalRIE_Depression(String)
getNotified(AA1Monitor_Depres...
notifyGlobalRIE():boolean
triggerAction():void
getAction():String
printAction():void
toString():String
<<Java Class>> AA1Executor_Depression org.domain
executorID: String
commandID: String
actionID: String
adapterID: String
notification: boolean
AA1Executor_Depression(St...
getNotified(GlobalRIE):boolean
getCommand(String,String,St...
executeAction():void
applyChanges():void
toString():String
<<Java Class>> LocalRIE org.manager
<<Java Class>> AAEngine (default package)
<<Java Class>> Monitor org.manager
<<Java Class>> Executor org.manager
<<Java Class>> AADepression org.adapter
AADepression()
<<Java Class>> ActionImpl org.manager
<<Java Class>> Action org.manager
<<Java Class>> AA1Monitor_Depression org.domain
notification: boolean
depContext: String
voiceContext: String
voiceUrgeSelf: String
voiceUrgeOther: String
delusions: String
monitorID: String
sensorData: String
context: String[]
AA1Monitor_Depression(...
getNotified(AASensor):bo...
notifyLocalRIE():boolean
<<Java Class>> AASensor org.manager
<<Java Class>> ManagedResource org.manager
resource: Resource
eventList1: ArrayList<String>
eventList2: ArrayList<String>
eventList3: ArrayList<String>
<<Java Class>> GlobalRIE org.manager
map: HashMap<String,Stri...
actionID: String
<<Java Class>> Policy org.policy.parser
policyID: String
policyType: String
policyDescription: String
ref_aa: String
resourceID: String
<<Java Class>> PolicyFileHandler org.policy.parser
<<Java Class>> PolicyData org.manager
depression: ArrayList<String>
voice_act: ArrayList<String>
voice_urge_self: ArrayList<String>
voice_urge_other: ArrayList<String>
delusions: ArrayList<String>
context1: ArrayList<String>
context2: ArrayList<String>
actions: ArrayList<String>
node_id: String
name: String
recepient: String
subject: String
txtMessage: String
<<Java Interface>> IManagedResource org.interfaces
<<Java Interface>> ISensor org.interfaces
~aa1_rie 0..1
~aa1_executor 0..1
-aa1_executor 0..1
-aa1_rie 0..1
-monitor 0..1
-callAction-action 0..1
~monitor 0..1
-sensor 0..1
~sensor
~sensor4
~sensor2
~sensor3
~sensor5
~sensor1
~act_on_voice
~delusions
~voice_urge_harm_self
~mr
~voice_urge_harm_others
~depression_status
-mr 0..1
-gRIE 0..1
~globalRIE 0..1
-policy 0..1
-policySet 0..*
-policyData 0..1
-policy
-sensors 0..*

Fig. 6.7 UML Class Diagram of AA Engine

6.5.1.1 Constructing AA's Intelligent Manager Abstract Classes

To implement AA's generic Intelligent Manager (IM) components, abstract classes and methods are defined for the *Monitor*, *LocalRIE* and *Executor* components. These abstract classes can be instantiated to create multiple AAs IM as class-subclass relationship and share data member and methods by all subclasses. The *AAEngine* class define a main method *public static void main(String[] args)* where many AA class objects can be created for the main entry point of the program for execution. A policy inference engine is also implemented with Java to parse policies from XML document for AAs that is configured in policy specification Section (6.4.2) in this chapter. The code fragment in listing 6.6 shows the partial implementation of AAs abstract classes and methods.

```
/* AA Monitor abstract class that declared abstract methods. */
public abstract class Monitor {
   // declare abstract methods.
   public abstract boolean getNotified(AASensor s);
   public abstract boolean notifyLocalRIE();
   public abstract void setContext();
      ...
} // end monitor abstract class

/* AA Local RIE abstract class that declared abstract methods. */
public abstract class LocalRIE {
   // declare abstract methods.
   public abstract void triggerAction();
   public abstract boolean notifyGlobalRIE();
       ...
} // end local rie abstract class

/* AA Executor abstract class that declared abstract methods. */
public abstract class Executor {
   // declare abstract methods.
   public abstract boolean getNotified(GlobalRIE rie);
   public abstract void getCommand(String commandID, String actionID, String
       adapterID);
   public abstract void executeAction();
   public abstract void applyChanges();
       ...
} // end executor abstract classs

/* AAEngine class is the main entry point of the program because
when we start a java program, it looks for the main method.
Many objects can be created from classes for instantiation. */
public class AAEngine {
   public static void main(String[] args) {
      ...
      ...
   } // end main method
} // end AAEngine class
```

Listing 6.6 : Intelligent Manager Abstract Classess.

6.5.1.2 Constructing AA's Sensor

To create an AA sensor, we fist define interface *ISensor* that is a completely abstract class and provide default implementations of methods. The *AASensor* class is a concrete implementation of *ISensor* interface that observes changes in a list of managed resource objects. For example, the *AASensor* class define a method, *update(Resource resource, String eventData)* to set the object to be observe and this method method is used by *ManagedResource* class to notify sensor for any updates occur. The code fragment in listing 6.7 shows the implementation of these classes.

```
/* Defining ISensor interface. */
public interface ISensor {
   // method to update the sensor, used by managed resource.
   public void update(Resource resource, String eventData);
   public void retrieveEventData();
     ...
} // end ISensor interface

/* Concrete AASensor class that implements of ISensor interface
to handle events and notification. */
public class AASensor implements ISensor {
   private String sensorID, sensorType;
   private ManagedResource mr;
     ...
   // constructor initialise varaibles.
   public AASensor(String sensorID, String sensorType) {
       this.sensorID = sensorID;
       this.mr = new ManagedResource(new Resource("","",""));
           ...
   } // end constructor

   // called when a change has occurred in the state of the managed resource.
   @Override
   public void update(Resource resource , String eventdata) {
       System.out.println(sensorID + "␣has␣received␣event␣data␣from␣"
        + resource + ",␣event␣value:␣"+  eventdata);
       this.notification = true;
   } // end method

   // notifyMonitor() method handle notification to monitor by sensors.
     public boolean notifyMonitor() {
         if(notification == true) {
             System.out.println("Sensor␣" + sensorID + "␣notified␣AA's␣Monitor"
                 );
             return true;
     } // end if
    return false;
    } // end method

    // retrieveEventData() retrieve event data from monitor.
    public void retrieveEventData() {
       System.out.print("Event␣Data:␣[");
          for(int i=0; i< mr.getEvents().size(); i++) {
             eventData[i] = mr.getEvents().get(i);
             System.out.print("␣"+ eventData[i] + ",");
           } // end for loop
        System.out.println("]␣retrieved␣by␣sensors\n");
   } // end method
   ...
} // end class
```

Listing 6.7 : AA’ Sessor Interface and Class.

6.5.1.3 Constructing Managed Resource

The *ManagedResource* class defines a list of sensors object *ArrayList<ISensor> sensors* and provide methods to register *registerSensor(ISensor sensor)* and unregister sensors *unregisterSensor(ISensor sensor)*. The *ManagedResource* class also contains a method *notifySensor(String eventData)* that provide functionality to notify all the registered sensors if any change received from managed resource objects. The interface *IManagedResource* is defined that provide default implementations of methods and the *ManagedResource* class implement *IManagedResource* interface. The code fragment in listing 6.8 shows the *IManagedResource* interface and concrete implementation of *ManagedResource* class.

```
/* Defining IManagedResource interface. */
public interface IManagedResource {
   //methods to register and unregister sensors.
   public void registerSensor(ISensor sensor);
   public void unregisterSensor(ISensor sensor);
   //method to notify sensors of changes.
   public void notifySensor(String eventData);
   //method to get updates from managed resource.
   public Object getUpdate(ISensor obj);
      ...
} // end interface

/* Concrete ManagedResource class that implements IManagedResource interface.
    */
public class ManagedResource implements IManagedResource {
    private ArrayList<ISensor> sensors;
    private Resource resource;
       ...
    //constructor initialise varaibles.
    public ManagedResource(Resource resource) {
      this.resource = resource;
      this.sensors = new ArrayList<ISensor>();
         ...
    } // end constructor

   // add sensor to the managed resources.
   @Override
    public void registerSensor(ISensor sensor) {
       sensors.add(sensor);
       System.out.println(sensor + "␣has␣started␣sensing␣event␣from␣" +
           resource);
    } // end method

   // remove sensor from the managed resources.
   @Override
   public void unregisterSensor(ISensor sensor) {
      sensors.add(sensor);
       System.out.println(sensor + "␣has␣started␣sensing␣event␣from␣" +
           resource);
   } // end method

   // notify all the registered sensors.
   @Override
   public void notifySensor(String eventData) {
      for(ISensor sensor : sensors) {
         sensor.update(resource, eventData);
      } // end for loop
   } // end method

   // set triggered event from manage resource.
    public void setEvent(String selectedEvent) {
       this.eventData = selectedEvent;
       this.eventList.add(eventData);
       System.out.println("\n" + resource + "␣has␣event␣value:␣" + this.
           eventData + "\n");
       notifySensor(eventData);
    } // end method
      ...
} // end ManagedResource class
```

Listing 6.8 : MR Interface and Class.

6.5.1.4 Construction Global RIE Abstract Class

The *GlobalRIE* class defines an abstract array list method *ArrayList<String> actions* to store all the actions coming from the AAs and provide methods to define decision as shown in the code listings 6.9.

```
/* GlobalRIE class implementation. */
public abstract class GlobalRIE {
   //declare abstract field and method.
   private abstract ArrayList<String> actions;
   public abstract void decision();
      ...
} // end abstract class
```

Listing 6.9 : GlobalRIE Abstract Class.

6.5.1.5 Creating PolicyElement Class.

The *PolicyElement* class define the accessor and mutator methods to store the individual elements and attributes parsed from the XML-based policy file, which is partially shown in listing 6.10.

```
/* the Policy class defines the accessor and mutatotr method for all
policy elements and attributes represented in XML document. */
public class PolicyElement {
/* declare fields for policy document elements */
 private String policyID, policyType, policyDescription, ref_aa;
 private String resourceID, resourceType, ref_sensor, ref_event;
  ...
 private Object AAPolicy, AA, ManagedResourceList, ManagedResource;
  ...
 private ArrayList<Object> Sensor;
  ...

 /* constructor to initilise the fields */
 public Policy() {
  listOfActions = new ArrayList<Object>();
   ...
 }

 public String getAdapterID() {
  return adapterID;
 }

 public void setAdapterID(String adapterID) {
  this.adapterID = adapterID;
 }

 public String getAdapterType() {
  return adapterType;
 }

 ...
 ...
} // end PolicyElement class
```

Listing 6.10 : Defining Policy Element Class.

6.5.1.6 Creating PolicyHandler Class.

The *PolicyHandler* class defines a set of callback handler methods that can be invoked when events occur during parsing. It reads the XML file and creates a list of object items from the entries in the XML file, which is partially shown in the code listing 6.11.

```
public class PolicyFileHandler extends DefaultHandler{
 //List to hold policies objects
 private ArrayList<Policy> policySet;
 private Policy policy;
 private String tmpValue;

 public PolicyFileHandler() {
  policyList = new ArrayList<Policy>();
  policy = new Policy();
  parseDocument();
 }

 public void parseDocument() {
  File cvPolicy = new File("./src/policy.xml");
  SAXParserFactory factory = SAXParserFactory.newInstance();
  try {
   SAXParser parser = factory.newSAXParser();
   parser.parse(cvPolicy, this);
  } catch(ParserConfigurationException e) {
   System.out.println("ParserConfig␣error");
```

```
    } catch(SAXException e) {
   System.out.println("SAXException␣:␣xml␣not␣well␣formed");
  } catch(IOException e) {
   System.out.println("IO␣error");
  }

 }//End parseDocument method

 //Triggered when the start of tag is found.
 @Override
 public void startElement(String uri, String localName, String elementName,
     Attributes attributes)
 throws SAXException {
  if(elementName.equalsIgnoreCase("AAPolicy")) {
  // if current element is Policy, Create a new policy and put it in map
  // clear tmpValue on start of element
   policy = new Policy();
   String policyID = attributes.getValue("policyID");
   String policyType = attributes.getValue("policyType");
   String policyDescription = attributes.getValue("policyDescription");
   String ref_aa = attributes.getValue("ref_aa");

   policy.setPolicyID(policyID);
   policy.setPolicyType(policyType);
   policy.setPolicyDescription(policyDescription);
   policy.setRefAA(ref_aa);

  } // end if statements

 } // end startElement Method

 ...
 ...
} // end PolicyFileHandler class
```

Listing 6.11 : Defining Policy Handler Class.

After the above Java implementation of necessary program classes, methods and policies, the MG scores from eGRiST assessment dataset are set to the AAs sensors as generated event variable for the managed resources and observe their output results such as the intervention decision. After executing the program, the output produced by the program indicates that the system is feasible and performing all of the operations correctly as expected.

6.6 Testing And Evaluation

This subsection presents an experimental testing and evaluation of the proposed AAs operation and policy model based on the collected and synthesised data from eGRiST, discussed in Section (6.3). Due to the lack of freely available open source autonomic system API, we implement a prototype Java console-based application of the AA engine to validate the feasibility of our proposed method introduced in Chapter (5).

To test and evaluate the system feasibility, we have created three instances of AAs, *AADepression* class as AA1, *AASuicide* class as AA2 and *AASocial* class as AA3 and their intelligent manager component classes such as *AA1Monitor_Depression* class, *AA1LocalRIE_Depression* class, and *AA1Executor_Depression* class, that inherits AA design pattern classes and method. We also implement *GlobalRIE* class that evaluates actions and trigger final intervention. The policy elements are parsed and instantiated as Java objects in order to test a range of input and output according to the eGRiST risk assessment scenario illustrated in Section (6.3.2. Several event data are input to the AAs sensors as environment variable and observe their output results such as the intervention decision.

Due to the limited space, we do not show the full implementation of all classes. Due to space limitation, we only show the partial code in this section. However, the complete implementation of AADepression adapter, AASuicide adapter and AASocial adapter and the related configured policy document is available online at the URL https://github.com/nhussain920/thesis_source_code.

The following sub-sections discuss the partial Java program implementation for *AADepression* adapter and its intelligent manager component classes: *AA1Monitor_Depression*, *AA1LocalRIE_Depression*, and *AA1Executor_Depression* and shows the relevant input and output after executing the program.

6.6.1 Creating AADepression Class and Sensor Instances

As shown in the code fragment below, we have defined *AADepression* class where managed resource instances and sensor instances are created as 'sensor1', 'sensor2', 'sensor3', 'sensor4', and 'sensor5' and registered with associated managed resources 'depression_status', 'act_on_voice', 'voice_urge_harm_self', 'voice_urge_harm_others' and 'delusions', respectively. Then, the MG variables (as shown in eGRiST scenarios) are set by invoking *setEvent()* method as: 'depression_status = 0.6', 'act_on_voice = 0.6', 'voice_urge_harm_self', 'voice_urge_harm_others = 0.4' and 'delusions = 0.7'. Each sensor observes and retrieved these events variables during the execution the program. The Java code of *AADepression* class is partially shown in listing **??**.

```
/*AA1Depression class implementation. */
public class AADepression {
 // delare variables and class instances as fields.
 ManagedResource depression_status;
 ManagedResource act_on_voice;
 ManagedResource voice_urge_harm_self;
 ManagedResource voice_urge_harm_others;
 ManagedResource delusions;
 AASensor sensor, sensor1, sensor2, sensor3, sensor4, sensor5;

 // declare constructor to initialise fields .
 public AADepression() {
  // initialise managed resource instances.
```

```
    depression_status = new ManagedResource(new Resource("mr1", "Depression␣
        Status", "s1"));
    act_on_voice = new ManagedResource(new Resource("mr2", "Act␣On␣Voice", "s2")
        );
    voice_urge_harm_self = new ManagedResource(new Resource("mr3", "
        voice_urge_harm_self", "s3"));
    voice_urge_harm_others = new ManagedResource(new Resource("mr4", "
        voice_urge_harm_others", "s4"));
    delusions = new ManagedResource(new Resource("mr5", "delusions", "s5"));

    // creating sensors instances.
    sensor1 = new AASensor("S1", "AA1␣Depression␣Sensor");
    sensor2 = new AASensor("S2", "AA1␣Act␣on␣Voice␣Sensor");
    sensor3 = new AASensor("S3", "AA1␣Voice␣Urge␣Self␣Sensor");
    sensor4 = new AASensor("S4", "AA1␣Voice␣Urge␣Other␣Sensor");
    sensor5 = new AASensor("S5", "AA1␣Delusion␣Sensor");

    // registering sensors with managed resources.
    depression_status.registerSensor(sensor1);
    act_on_voice.registerSensor(sensor2);
    voice_urge_harm_self.registerSensor(sensor3);
    voice_urge_harm_others.registerSensor(sensor4);
    delusions.registerSensor(sensor5);

    // setting environment event variables.
    depression_status.setEvent("first-diagnosis");
    act_on_voice.setEvent("0.6");
    voice_urge_harm_self.setEvent("0.2");
    voice_urge_harm_others.setEvent("0.4");
    delusions.setEvent("0.7");
    ...
    ...
  } // end constructor
} // end class
```

Listing 6.12 : AA1 Depression Class.

6.6.2 Creating AA1Monitor_Depression Class and Instances

In the code fragment below, we have created *AA1Monitor_Depression* class that extends abstract *Monitor* class. The concrete implementation of the *setContext()* method defines a new context variable for each monitored event according to the policy. The class definition is partially shown in the program code listing 6.13.

```
/* AA1Monitor_Depression class implementation that extends abstract Monitor
    class. */
public class AA1Monitor_Depression extends Monitor {
 // delare class instances and variables as fields.
 private ManagedResource mr;
 private String depContext, voiceContext, voiceUrgeSelf,
 voiceUrgeOther, delusions;
  ...
 // declare constructor to initialise fields.
 public AA1Monitor_Depression(String monitorID) {
  mr = new ManagedResource(new Resource("","",""));
   ...
   ...
 } // end constructor

 // set new context variables according to the condition evaluation.
 public void setContext() {
  for(int i=0; i<mr.getEvents().size(); i++) {
   if(mr.getEvents().contains(policyData.getDepressionData().get(i))) {
    depContext = "low";
    context[0] = depContext;
    System.out.println("New␣context␣variable␣set␣for␣'Depression'␣Node␣:␣" +
        depContext);
   } // end if

   if(mr.getEvents().contains(policyData.getVoiceActData().get(i))) {
    voiceContext = "likely";
    context[1] = voiceContext;
    System.out.println("New␣context␣variable␣set␣for␣'Act␣On␣Voice'␣Node:␣" +
        voiceContext);
   } // end if
    ...
    ...
  } end for loop
 } // end setContext() method
 ...
 ...
} // end class
```

Listing 6.13 : AA1Monitor_Depression Class.

6.6.3 Creating AA1LocalRIE_Depression Class and Instances

In the code fragment below, we have created *AA1LocalRIE_Depression* class that extends the abstract *LocalRIE* class. The concrete implementation of the *triggerAction()* method evaluates the context variables and trigger actions according to the rule condition in the policy, which is partially shown in listing 6.14.

```
/* AA1LocalRIE_Depression class implementation that extends abstract LocalRIE
    class. */
public class AA1LocalRIE_Depression extends LocalRIE {
 // declare class instances and variables as fields.
 private AA1Monitor_Depression monitor;
 private String rieID;
  ...
 // declare constructor to initialise fields.
 public AA1LocalRIE_Depression(String rieID) {
  this.rieID = rieID;
   ...
 } // end constructor
 // define triggerAction() method to trigger action context true.
 public void triggerAction() {
  int i = 0;
  for(i=0; i<monitor.getContext().length; i++) {
   if( monitor.context[i].equals(policyData.getExpression1().get(i)) &&
    monitor.context[++i].equals(policyData.getExpression1().get(i)) &&
    monitor.context[++i].equals(policyData.getExpression1().get(i)) &&
    monitor.context[++i].equals(policyData.getExpression1().get(i)) &&
    monitor.context[++i].equals(policyData.getExpression1().get(i))) {
    System.out.println("Context␣1␣is␣true");
    System.out.println("Context␣1␣is␣true");
    System.out.println("Trigger␣action␣template␣open_url");
    AA1LocalRIE_Depression.trigger_action = "a1";
   } // end if
   ...
   ...
  } // end for loop
 } // end triggerAction() method
 ...
 ...
} // end class
```

Listing 6.14 : AA1LocalRIE_Depression Class.

6.6.4 Creating Global RIE Class and Instances

In the code fragment below, we have created *GlobalRIE* class that defines *decision()* method to evaluate actions coming from AAs and decide what actions to be executed according to the rule condition in the global policy, which is partially shown in listing 6.15:

```
/* GlobalRIE class implementation. */
public class GlobalRIE {
 // declare class instances and variables as fields.
 private ArrayList<String> actions;
 private AA1LocalRIE_Depression aa1_rie;
  ...
 // declare constructor to initilise fields.
 public GlobalRIE(String rieID) {
  this.actions = new ArrayList<String>();
   ...
 } // end constructor

 // declare addActions() method that add triggered actions coming from AAs.
 public void addActions() {
  map.put(aa1_rie.getAction(), aa1_rie.toString());
  map.put(aa2_rie.getAction(), aa2_rie.toString());
  map.put(aa3_rie.getAction(), aa3_rie.toString());
   ...
 } // end addActions() method

 // declare decision() method confirms actions to be execute based on the
     policy.
 public void decision() {
  for (HashMap.Entry<String, String> entry : map.entrySet()) {
   actionID = entry.getKey();
   adapterID = entry.getValue();

   if(map.containsKey(actionID) && map.containsValue(adapterID)) {
    System.out.println("Execute␣action:␣" + actionID);
    aa1_executor.getCommand("cmd1", actionID, "AA1Depression");
   } // end if

   else if(map.containsKey("aa-rie1") && map.containsValue("a1")) {
    System.out.println("Execute␣action:␣" + actionID);
   } // end else if
    ...
  } // end for loop
 } // end decision() method
  ...
  ...
} // end class
```

Listing 6.15 : Global RIE Class that map AA1's triggered actions.

6.6.5 Creating AA1Executor_Depression Class and Instances

In the code fragment below, we have created *AA1Executor_Depression* class that extends the abstract *Executor* class. The concrete implementation of the *executeAction()* method executes the actions confirmed by global RIE and *applyChanges()* method sent the intervention to the social-collaborative network, which is partially shown in listing 6.16.

```
/* AA1Executor_Depression class implementation that extends abstract Executor
    class. */
public class GlobalRIE {
 // declare class instances and variables as fields.
 private ArrayList<String> actions;
 private AA1LocalRIE_Depression aa1_rie;
  ...
 // declare constructor to initilise fields.
 public GlobalRIE(String rieID) {
  ...
 } // end constructor

 // declare executeAction() method.
 public void executeAction() {
  String id = policy.node_id;
  String name = policy.name;
  String recepient = policy.recepient;
  String subject = policy.subject;
  String txtMessage = policy.txtMessage;
  String population_restriction = policy.population_restriction;
  action.sendEmail(id, name, recepient, subject, txtMessage,
      population_restriction);
   ...
 } // end executeAction() method
```

```
  // declare applyChanges() method.
  public void applyChanges() {

  } // end applyChanges() method
  ...
  ...
 } // end class
```

Listing 6.16 : AA1Executor_Depression Class.

6.6.6 Creating AAs Class Objects

In the code below, we have created *AAEngine* class that is used to define AA's class objects, *depression_aa*, *suicide_aa* and *social_aa*, which is partially shown in listing 6.17.

```
 /* AAEngine class impementation. */
 public class AAEngine {
  // declare main() method to run the program.
  public static void main(String[] args) {
   // creating AA's␣object.
␣␣␣AADepression␣depression_aa␣=␣new␣AADepression();
␣␣␣AASuicide␣suicide_aa␣=␣new␣AASuicide();
␣␣␣AASocial␣social_aa␣=␣new␣AASocial();
␣␣}␣//␣end␣main()␣method
␣}␣//␣end␣class
```

Listing 6.17 : Creating AAs Object in AAEngine Class.

6.6.7 Program Output

After the implementation of necessary program classes, methods and policies, the collected MG scores from eGRiST assessment questions are set to the AAs sensors as generated event variable and observe their output results such as the intervention decision. After executing the program, as expected, the output of the program indicates that the system is feasible and performing operation correctly. Due to the limited space, we only show the AA1 output in this chapter.
After executing the program, as expected, the output result of AA1 sensors indicates that for any possible events, sensors are capable of observe the resources' states correctly, which is partially shown in listing 6.18.

```
// sensors sensing events from managed resoures.
Sensor S1, Sensor Type AA1 Depression Sensor has started sensing event from
    Resource ID: mr1,
type: Depression Status, ref sensor: s1
Sensor S2, Sensor Type AA1 Act on Voice Sensor has started sensing event from
    Resource ID: mr2,
...
...
// sensors retrieved events data.
```

```
Resource ID: mr1, type: Depression Status, ref sensor: s1 has event value:
    first-diagnosis
S1 has received event data from Resource ID: mr1, type: Depression Status, ref
    sensor: s1,
event value: first-diagnosis
...
...
```

Listing 6.18 : AA1 Sensor Output.

The output of AA1 Monitor for Depression node is partially shown in listing listing 6.19.

```
// sensors notified monitor.
Sensor S1 notified AA's␣Monitor
Monitor␣received␣notification␣from␣sensor:␣Sensor␣S1,␣Sensor␣Type␣AA1␣Sensor␣1
Sensor␣S2␣notified␣AA's Monitor
Monitor received notification from sensor: Sensor S2, Sensor Type AA1 Sensor 2
...
...
// sensors data added to monitor.
Senosr's␣Data␣[first-diagnosis,␣0.6,␣0.2,␣0.4,␣0.7]␣added␣to␣Monitor

//␣new␣context␣variables␣set␣by␣monitor.
New␣context␣variable␣set␣for␣'Depression'␣Node␣:␣low
New␣context␣variable␣set␣for␣'Act On Voice'␣Node:␣likely
...
...
```

Listing 6.19 : AA1 Monitor Output.

The output of AA1 Local RIE for Depression node is partially shown in listing 6.20.

```
// monitor notified local RIE.
Monitor m1 notified AA's␣Local␣RIE
Local␣RIE␣received␣notification␣from␣Monitor:␣Monitor␣m1
//␣context␣1␣evaluated␣true.
Context␣1␣true
//␣send_email␣action␣triggered␣which␣id␣is␣a1.
Trigger␣action␣template␣send_email
Triggered␣Action␣ID:␣a1␣
```

Listing 6.20 : AA1 Local RIE Output.

The output of Global RIE is partially shown in listing 6.21.

```
// local RIE notified global RIE.
Local RIE: aa-rie1 notified Global RIE
Local RIE: aa-rie2 notified Global RIE
Local RIE: aa-rie3 notified Global RIE

// global RIE received notification from global RIE.
Global RIE received notification from: AA1 Local RIE: aa-rie1
Global RIE received notification from: AA2 Local RIE: aa-rie2
Global RIE received notification from: AA3 Local RIE: aa-rie3
// global RIE decided action a1 to be executed and sent the command to AA1's␣
    executor.
```

```
Execute␣action:␣a1
//␣global␣RIE␣decided␣action␣a1␣to␣be␣executed␣and␣sent␣the␣command␣to␣AA2's
    executor.
Execute action: a1
// global RIE decided action a1 to be executed and sent the command to AA3's␣
    executor.
Execute␣action:␣a1
AA1Depression␣adapter␣recieved␣command:␣cmd1␣from␣Global␣RIE␣to␣execute␣action:
    ␣a1
AA2Siucide␣adapter␣recieved␣command:␣cmd2␣from␣Global␣RIE␣to␣execute␣action:␣a1
AA3Social␣adapter␣recieved␣command:␣cmd3␣from␣Global␣RIE␣to␣execute␣action:␣a1
```

Listing 6.21 : Global RIE Output.

The output of AA1 Executor for Depression that execute some actions is partially shown in listing 6.22.

```
// global RIE notified executor.
Global RIE ID: GlobalRIE notified AA's␣Executor
//␣executor␣received␣notification␣from␣global␣RIE.
Executor␣received␣notification␣from:␣GlobalRIE
//␣executing␣email␣action.
Sending␣email␣...
Email␣ID:␣n1,␣Person␣Name:␣John,␣Email:␣abc@example.come,␣Subject␣line:␣medical
    ␣advice,
Message:␣We␣recommended␣you␣to␣seek␣help...,␣Population␣Category:␣older␣adults
```

Listing 6.22 : AA1 Executor Output.

In the eGRiST program implementation and testing above, we first implemented generic AA Engine, including concrete implementation of sensor interface and intelligent manager loop and policy parser in Java as shown in above section (6.5). Then, we have extended the *AA Engine* with the implementation of *AA1* classes for Depression node, which is partially shown in above section (6.6.1. Similarly, *AA2* and *AA3* classes are implemented (code is available at https://github.com/nhussain920/thesis_source_code) for Suicide and Social concept node of eGRiST system. The environment variables are set as input events for each managed resource that are associates with each AA and observed the outputs of each intelligent manager component separately. The program implementation of AA engine and the integrated AAs into eGRiST, AA1, AA2 and AA3, which contains many abstract and concrete classes, interfaces and methods, which are a very large amount of code and the program output. Thus, due to the limited space, we have partially shown few essential classes and the output but omitted many lines of the code from the main text. Although, our current program simulation has produced the output as expected, unfortunately, it is difficult to comprehensively determine whether a program validation is sufficient. Because, a realistic large-scale software-development context and testing scenarios, the program simulation may have various bottlenecks such as program bugs, insufficient performance, optimisation, memory and resource leaks under different circumstances. Therefore, we plan to develop more sophisticated program code, which could be easy to analysis, sufficiently debugged, reusable and scalable to provide relevant indicators to reconfigure the managed system. Finally, we expect to explore additional experiment in different contexts such as cyber-physical smart cities or embedded systems in which the reconfiguration scenarios and goals are different.

6.7 Conclusion

This chapter presented a flexible and consistent policy model and specification language for supporting the policy-based interactions and decision making between AAs. This is a modest contribution to overcome the limitation of existing fixed coded autonomic computing issues, especially the self-management and adaptive characteristics to reduce human agent intervention. Our policy model incorporates ECA rules as shared knowledge, which is enforced by the rule inference engine within the management components of the AAs. We proposed PSL and PSD that are used to configure and validate the policy, respectively. We described a prototype implementation, testing and validation of the proposed policy-based autonomic control service through a real-world mental health decision support system case study, namely eGRiST. We developed a Java-based program simulation of integrated AAs into eGRiST system, and the input and output test established according to eGRiST assessment questions by executing program. The output result proved that the system is feasible, consistent and valid on given input values that produced the expected results.

Some general requirements that any policy design and representation should satisfy regardless of its field of applicability and expressiveness to handle the wide range of policy requirements, simplicity to ease the policy definition, enforceability to various platforms, scalability to ensure adequate performance. More work is needed on implementing and testing our AA policy model in different domains in terms of the general requirements of the policy design and representation. This would provide the autonomic system a more robust way to evaluate system operation in various environments.

CHAPTER 7

Conclusion and Future Work

This chapter presents a summarised discussion of the main thesis contributions and draws some conclusions from this study and highlights some future works, which may further improve the current research outcomes.

7.1 Contributions

This thesis attempts to design and develop a self-managed and adaptive socio-cyber-physical collaborative system by utilising social computing and autonomic computing approach. We addressed the problem of how socio-cyber-physical resources can be semantically represented and can interact, collaborate and make a decision without human interventions. Afterwards, we proposed GASCF and AA architecture and explored how different AAs should be deployed across the IT infrastructure and how policy could manage and automate the AAs behaviour for a collaborative decision-making process within a socio-cyber-physical environment. Finally, we discussed the implementation and evaluation of the proposed approach in real-world healthcare systems, name GRiST online healthcare service and GRiST CDSS. In summary, the main contributions of this thesis and the novel ideas are as follows:

- Social computing methodology has been studied and used with encouraging results in developing collaborative tools and applications in different domains. However, existing collaboration approaches do not consider how cyber-physical objects can establish and maintain their social relations, interactions and communications to achieve a shared and collaborative decision-making goal based on their event status. Existing social networking applications developed for general networking purposes such as to keep in touch with family and friends, or online conversations only [186] and lack of semantic structure for supporting socio-cyber-physical relation and collaboration abilities. A few research evidenced that people not only connect to each other directly, but they also connect through shared objects or artifacts [24]. In the literature, several social agent-based solutions have been discussed as a set of micro-services to handle digital traces of human activities, respecting both users' privacy and the business models of companies to build personalised services or applications in the web [10]. Unfortunately, these social agent model are not adequate to represent and act on behalf of the cyber-physical resources in a social environment.

To overcome this limitation, in Chapter (4), we investigated the impacts and benefits of employing agent-based social networking approach with semantic ontology integration for supporting human and nonhuman (e.g. cyber-physical resources) interactions and collaboration task. First, we proposed a software agent-centric semantic social-collaborative network (SSCN) where socio-cyber-physical resources represented as a social graph. These nonhuman resources or social artifacts cannot interact and communicate with each other by themselves. Therefore, the software agent component was integrated into the framework that provides the functionality to manage and maintain resources activities and actions. The OWL-based ontology model also introduced that provides a common semantic knowledge framework to represent concept, properties and relation of a domain of discourse, and also improve data interoperability, accessibility and information inference from the hidden facts.

To validate the proposed approach, we discussed GRiST online healthcare service and constructed a domain-centric ontology to represent domain concepts (i.e., human, smart devices, web resources and medical artifacts), properties and relations among them. We performed ontology reasoning using existing reasoner to check the correct syntax and consistency of the ontology. Several SWRL and SPARQL queries were applied to test the correct functioning of the ontology. The query results show that semantic representation of healthcare network provides efficient interconnection and discovery of the hidden information from the resources interactions.

The socio-cyber-physical collaborative system rapidly becomes brittle and unmanageable due to the growing complexity, heterogeneity of the cyber-physical resources and their social dynamics. Therefore, it is a difficult task and error-prone for traditional software agents model to cope with this large and uncertain situation within the social-collaborative environment. To provide a solution to this problem, we investigated the autonomic computing approach to create and leverage intelligent agent into the system for self-management capabilities.

- Autonomic computing was introduced to manage complex computer applications and systems without human intervention [106]. The intelligent control loop (monitor, analysis, plan, execute and knowledge) is the core concept of autonomic computing [56]. Most of the existing autonomic computing approaches have been tightly coupled with the domain-specific applications for which they have been developed. A few have provided frameworks that focused on the self-adaptive system, but they are not adequate for autonomous socio-cyber-physical resource collaboration. To fulfil this gap, in Chapter

(5), we proposed GASCF and AA architecture to make the system self-managed by enabling autonomous monitoring of resource states, analysis the situations, trigger and execute actions in a real-time according to the rules in the policy. This GASCF framework focuses on the higher-level social context, and the use of AAs communicates and collaborates with each other through a social network for collaborative decision making.

Our GASCF and AA model has several significant advantages over existing agent-based approaches with regards to socio-cyber-physical autonomic system development. It is universal enough to be re-configured for use with any legacy or future IT system whose behaviour can be controlled by the policy without disrupting the normal system operation. The decisions taken by the AAs are based on a rich and flexible set of high-level ECA rule-based policies that are unavailable in existing cyber-physical solutions or multiagent-based systems. Moreover, each AA has a local rule inference engine that triggers single action, whereas a global rule inference engine collates output actions coming from AAs to make a final decision, which is new to the autonomous agent model. Both rule inference engines constantly communicate with shared knowledge source to fetch rule for action intervention whenever needed.

As practical applicability and validity of our proposed approach, we discussed how GASCF and AAs can be applied to extended GRiST domain to implement a self-managed and adaptive healthcare system. We have shown that AAs enables continuous monitoring of a patient's health condition, the collection of patient data, and triggers some actions such as notification alarms when critical conditions are detected so that doctors or carers can instantly react to them. This helps to reduce the burden of data input required by the older adults themselves but also provide a mechanism for monitoring their safety at home on a continuous basis.

However, the current challenges of autonomic computing research are related to designing a general-purpose, flexible and consistent policy model and specification language for managing AAs behaviour within the application domain. In general, low-level decision-making logic is usually encapsulated by Event-Condition-Action (ECA) rules as a fixed coded within the autonomic computing knowledge [22]. The common uses of fixed coded policies are inadequate and inflexible to dynamically change contexts [26]. To address this problem, we conducted a comprehensive study within the area of policy specification language for modelling general-purpose policies autonomic system.

- Policy-based management has emerged as an attractive approach for flexible and dynamically controlled systems, services and network management [143]. The use of a policy-based approach has received considerable attention in autonomic computing for the management of large and distributed IT environments. There are a number of approaches introduced to define the policies with accompanying policy languages, ranging from logic-based languages, special-purpose policy languages, generic rule-based (if-then-else) formats, ontology-based approaches, and so on [36]. However, there is no well-defined policy model compatible with the autonomic management of the socio-cyber-physical collaborative system.

We address this challenge in Chapter (6) by designing and implementing an ECA rule-based policy model and specification language, represented in XML format to control AAs behaviour at run-time. At a core, we proposed Policy Schema Definition (PSD) language that allows us to construct policy according to the schema structure. A general-purpose Policy Specification Scripts (PSS) also discussed that are easily reusable for domain-specific policy configuration. Finally, we implemented a generic AA Engine API in Java programming language and extended the classes and interface to create AAs, integrated into the eGRiST system. We validated the effectiveness of our AA policy model by implementing it to a web-based mental health risk decision support system, called eGRiST. To test the functionality of the system, we input synthesised data from the eGRiST as events parameter to the AAs and observed the output actions by running the system. The results show that risk assessment interventions are correctly triggered and feasible according to the AAs policy configuration, with appropriate system functionality and configuration capability.

7.2 Future Work

Dealing with uncertainty in a socio-cyber-physical system integrated with a large number of heterogeneous resources (i.e., mobile devices, sensors, cameras, wearable devices, etc.) is a challenging area for future research. We intend to improve the AAs analysing capability with the machine learning techniques to deal with predicting uncertain context information more accurately and self-adaptive decision-making. In addition to tackle heterogeneity, these resources will need to be consistently and formally represented through suitable abstraction technologies (i.e., ontology). However, how to use Semantic Web ontology to formally define context-aware resources and how to exploit it within the AAs intelligent manager's loop are a future research direction. Moreover, our proposed approach also need to be tested in other domain to evaluate the scalability and performance of the system.

REFERENCES

[1] Abbattista, F., Calefato, F., Gendarmi, D., and Lanubile, F. (2008). Incorporating Social Software into Distributed Agile Development Environments. In *23rd IEEE/ACM International Conference on Automated Software Engineering*, volume 1, pages 46–51. IEEE Press.

[2] Afzal, B., Umair, M., Asadullah Shah, G., and Ahmed, E. (2019). Enabling IoT platforms for social IoT applications: Vision, feature mapping, and challenges. *Future Generation Computer Systems*, 92:718–731.

[3] Aggarwal, P. K., Nigam, P., and Shrivastava, V. (2016). Self Controlled Traffic Management Using Autonomic System. In *3rd International Conference On Computing for Sustainable Global Development (INDIACom)*, pages 3813–3815, New Delhi. IEEE.

[4] Ahuja, K. and Dangey, H. (2014). Autonomic Computing: An Emerging Perspective and Issues. *Proceedings of the 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques, ICICT 2014*, pages 471–475.

[5] Alaya, M. B. and Monteil, T. (2012). FRAMESELF: A generic autonomic framework for self-management of distributed systems -Application on the self-configuration of M2M architecture using semantic and ontology. In *International Conference on Collaboration Technologies and Infrastructures (IEEE WETICE 2012)*. IEEE.

[6] Alirezaie, M., Renoux, J., Köckemann, U., Kristoffersson, A., Karlsson, L., Blomqvist, E., Tsiftes, N., Voigt, T., and Loutfi, A. (2017). An Ontology-based Context-aware System for Smart Homes: E-care@home. *Sensors*, 17(7):1–23.

[7] Almeida, M., Souza, R., and Fonseca, F. (2018). Semantics in the Semantic Web: A Critical Evaluation. *Knowledge Organization*, 38(3):187–203.

[8] Almomen, S. and Menascé, D. A. (2011). An Autonomic Computing Framework for Self-managed Emergency Departments. *HEALTHINF 2011-Proceedings of the International Conference on Health Informatics*, pages 52–60.

[9] Alti, A., Lakehal, A., Laborie, S., and Roose, P. (2016). Autonomic semantic-based context-aware platform for mobile applications in pervasive environments. *Future Internet*, 8(4):1–26.

[10] Alvertis, I., Biliri, E., Lampathaki, F., and Askounis, D. (2018). Social Agents to Enable Pervasive Social Networking Services. *Journal of Theoretical and Applied Electronic Commerce Research*, 13(3):50–84.

[11] Aman, W. and Snekkenes, E. (2015). EDAS: An evaluation prototype for autonomic event-driven adaptive security in the internet of things. *Future Internet*, 7(3):225–256.

[12] Anthony, R. (2006). Generic Support for Policy-Based Self-Adaptive Systems. *17th International Conference on Database and Expert Systems Applications (DEXA'06)*, pages 108–113.

[13] Anthony, R. J. (2007). Policy-centric integration and dynamic composition of autonomic computing techniques. In *In Autonomic Computing, 2007. ICAC'07.*, pages 2–2. IEEE.

[14] Ardini, A., Hosseini, M., Alrobai, A., Shahri, A., Phalp, K., and Ali, R. (2014). Social computing for software engineering: A mapping study. *Computer Science Review*, 13-14(C):75–93.

[15] Arnaboldi, V., Conti, M., Passarella, A., and Dunbar, R. I. (2017). Online Social Networks and information diffusion: The role of ego networks. *Online Social Networks and Media*, 1:44–55.

[16] Ashraf, Q. M., Yeoh, C. Y., Khalaf, A. A., Al-Haddad, A., Habaebi, M. H., Abdullah, W. R. W., and Yahya, M. R. (2017). Autonomic Internet of Things for Enforced Demand Management in Smart Grid. *American Journal of Data Mining and Knowledge Discovery*, 2(2):69–75.

[17] Atzori, L., Iera, A., Morabito, G., and Nitti, M. (2012). The Social Internet of Things (SIoT) – When Social Networks meet the Internet of Things: Concept, Architecture and Network Characterization. *Computer Networks*, 56(16):3594–3608.

[18] Ayala, I., Amor, M., and Fuentes, L. (2013). Self-configuring agents for ambient assisted living applications. *Personal and Ubiquitous Computing*, 17(6):1159–1169.

[19] Baader, F., Horrocks, I., and Sattler, U. (2005). Description logics as ontology languages for the semantic web. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2605 LNAI:228–248.

[20] Badr, N., Taleb-Bendiab, A., and Reilly, D. (2004). Policy-Based Autonomic Control Service. In *Policy*, pages 99–102.

[21] Bailey, C., Chadwick, D. W., and Lemos, R. D. (2011). Self-adaptive authorization framework for policy based RBAC/ABAC models. In *IEEE 9th International Conference on Dependable, Autonomic and Secure Computing, DASC 2011*, pages 37–44.

[22] Bailey, J., Poulovassilis, A., and Wood, P. T. (2002). An event-condition-action language for XML. *Proceedings of the eleventh international conference on World Wide Web - WWW '02*, page 486.

[23] Beach, A., Gartrell, M., Xing, X., Han, R., Lv, Q., Mishra, S., and Seada, K. (2010). Fusing Mobile, Sensor, and Social Data To Fully Enable Context-Aware Computing. *Proceedings of the 23nd annual ACM symposium on User interface software and technology - UIST '10*, page 105.

[24] Begel, A. and Deline, R. (2009). Codebook: Social Networking Over Code. In *31st International Conference on Software Engineering - Companion Volume*, pages 263–266, Bancouver, BC. IEEE.

[25] Begel, A., DeLine, R., and Zimmermann, T. (2010). Social media for software engineering. In *Proceedings of the FSE/SDP workshop on Future of software engineering research - FoSER '10*, pages 33–38, NY. ACM.

[26] Belhaj, N., Belaïd, D., and Mukhtar, H. (2017). Self-adaptive Decision Making for the Management of Component-Based Applications. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, pages 570–588. Springer.

[27] Bennett, M. and Baclawski, K. (2017). The Role of Ontologies in Linked Data, Big Data and Semantic Web Applications. *Applied Ontology*, 12(3-4):189–194.

[28] Bergstrom, A., Clark, R., Hogue, T., Iyechad, T., Miller, J., Mullen, S., Perkins, D., Rowe, E., Russell, J., Simon-Brown, V., Slinski, M., Snider, A. B., and Thurston, F. (1995). Collaboration framework: Addressing community capacity. Technical report, National Network for Collaboration, Fargo, ND.

[29] Berners-Lee, T., Fielding, R., and Masinter, L. (2005). Uniform resource identifier (URI): Generic syntax.

[30] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). THE SEMANTIC WEB. *Scientific American*, 284(5):28–37.

[31] Beyerlein, M. M. and Harris, C. L. (2002). Critical Success Factors in Team-Based Organising A Top Ten List. *The Collaborative Work Systems Fieldbook*, page 3.

[32] Bhowmick, S., Mohania, M., Sakamuri, B. C., Madria, S., and Lane, K. P. (2002). A Model for XML Schema Integration. *E-Commerce and Web Technologies*, pages 193–202.

[33] Biamino, G. (2011). Modeling Social Contexts for Pervasive Computing Environments. In *8th IEEE Workshop on Context Modeling and Reasoning Modeling*, pages 415–420. IEEE.

[34] Boley, H. (2001). The Rule Markup Language: RDF-XML Data Model, XML Schema Hierarchy, and XSL Transformation. *International conference on Applications of Prolog*, pages 5–22.

[35] Bourdenas, T. and Sloman, M. (2010). Starfish: Policy Driven Self-Management in Wireless Sensor Networks. In *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pages 75–83.

[36] Bradshaw, J. M., Uszok, A., and Montanari, R. (2014). Policy-Based Governance of Complex Distributed Systems: What Past Trends Can Teach Us about Future Requirements. *Agile Computing*, pages 259–284.

[37] Breslin, J. and Decker, S. (2007). The Future of Social The Need for Semantics. *Ieee Internet Computing*, 5:86–90.

[38] Buckingham, C. D. (2002). Psychological cue use and implications for a clinical decision support system. *Medical Informatics and the Internet in Medicine*, 27(4):237–251.

[39] Buckingham, C. D., Adams, A., Vail, L., Kumar, A., Ahmed, A., Whelan, A., and Karasouli, E. (2015). Integrating service user and practitioner expertise within a web-based system for collaborative mental-health risk and safety management. *Patient Education and Counseling*, 98(10):1189–1196.

[40] Buckingham, C. D., Ahmed, A., and Adams, A. (2013). Designing multiple user perspectives and functionality for clinical decision support systems. In *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on*, pages 211–218.

[41] Calinescu, R. (2009). Resource-definition policies for autonomic computing. In *5th International Conference on Autonomic and Autonomous Systems, ICAS 2009*, pages 111–116. IEEE.

[42] Calinescu, R. and Kwiatkowska, M. (2009). Using Quantitative Analysis to Implement Autonomic IT Systems a software framework that achieves such adaptiveness for. In *IEEE 31st International Conference on Software Engineering*, pages 100–110.

[43] Cano, J., Rutten, E., and Cano, J. (2014). Coordination of ECA rules by verification and control Julio. In *International Conference on Coordination Languages and Models*, pages 33–48. Springer.

[44] Cardoso, J. and Pinto, A. M. (2015). The Web Ontology Language (OWL) and its Applications. *Encyclopedia of Information Science and Technology, Third Edition*, pages 7674–7683.

[45] Carstensen, P. H. and Schmidt, K. (1999). *Computer supported cooperative work: New challenges to systems design.*

[46] Cassandras, C. G. (2016). Smart Cities as Cyber-Physical Social Systems. *Engineering*, 2(2):156–158.

[47] Cataldo, M., Wagstrom, P. A., Herbsleb, J. D., and Carley, K. M. (2006). Identification of coordination requirements: implications for the Design of collaboration and awareness tools. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work (CSCW 06)*, pages 353–363, New York. ACM.

[48] Catherine, M. and Lee, M. J. (2009). Pedagogical Responses to Social Software in Universities. In Stylianos, H. and Steven, W., editors, *Handbook of Research on Social Software and Developing Community Ontologies*, chapter ch023, pages 335–356. IGI Global.

[49] Challenger, M. (2012). The Ontology and Architecture for an Academic Social Network. *IJCSI International Journal of Computer Science Issues*, 9(2):22–27.

[50] Chan, H., Segal, A., Verma, D., Arnold, B., Giles, J., Agrawal, D., and Olshefski, D. (2003). An Approach to Policy Infrastructure for Self- Managing Systems. In *First Workshop on Algorithms and Architectures for Self-Managing System, Federated Computing Research Conference.*

[51] Chen, J. and Liu, J. (2014). Introduction: Social Computing and Social Networks. *Journal of Organizational Computing and Electronic Commerce*, 24(2-3):119–121.

[52] Chen, L., Lu, D., Zhu, M., Muzammal, M., Samuel, O. W., Huang, G., Li, W., and Wu, H. (2019). OMDP: An ontology-based model for diagnosis and treatment of diabetes patients in remote healthcare systems. *International Journal of Distributed Sensor Networks*, 15(5).

[53] Chen, R. C., Hendry, and Huang, C.-Y. (2016). A Domain Ontology in Social Networks for Identifying User Interest for Personalized Recommendations. *Journal of Universal Computer Science*, 22(3):319–339.

[54] Christopoulou, S. C., Kotsilieris, T., Anagnostopoulos, I., Anagnostopoulos, C. N., and Mylonas, P. (2017). vhMentor: An Ontology Supported Mobile Agent System for Pervasive Health Care Monitoring. *Advances in Experimental Medicine and Biology*, 989:57–65.

[55] Compton, M., Barnaghi, P., Bermudez, L., García-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., Huang, V., Janowicz, K., Kelsey, W. D., Le Phuoc, D., Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page, K., Passant, A., Sheth, A., and Taylor, K. (2012). The SSN ontology of the W3C semantic sensor network incubator group. *WebSemantics: Science,ServicesandAgentsontheWorldWideWe*, 17:25–32.

[56] Computing, A. (2006). An Architectural Blueprint for Autonomic Computing. Technical report, IBM Corporation.

[57] Cordoş, A.-A., Bolboacă, S., and Drugan, C. (2017). Social Media Usage for Patients and Healthcare Consumers: A Literature Review. *Publications*, 5(2):9.

[58] Damiani, F., Giannini, P., Ricci, A., and Viroli, M. (2011). A Calculus of Agents and Artifacts. In *International Conference on Software and Data Technologies*, volume 50, pages 124–136. Springer Berlin Heidelberg.

[59] Damianou, N., Dulay, N., and Lupu, E. (2001). The ponder policy specification language. In *Workshop on Policies for Distributed Systems and Networks*, pages 18–39.

[60] Das, S. K. (2016). Cyber-physical-social convergence in smart living: Challenges and opportunities. *2016 IEEE International Conference on Pervasive Computing and Communication Workshops, PerCom Workshops 2016*, page 1.

[61] Davy, S., Barrett, K., Serrano, M., Strassner, J., Jennings, B., and Van Der Meer, S. (2007). Policy Interactions and Management of Traffic Engineering Services Based on Ontologies. In *2007 Latin American Network Operations and Management Symposium - LANOMS 2007*, number June 2014, pages 95–105.

[62] De, S., Zhou, Y., Larizgoitia Abad, I., and Moessner, K. (2017). Cyber–Physical–Social Frameworks for Urban Big Data Systems: A Survey. *Applied Sciences*, 7(10):1–26.

[63] De Meo, P., Quattrone, G., and Ursino, D. (2010). Integration of the HL7 standard in a multiagent system to support personalized access to e-health services. *IEEE Transactions on Knowledge and Data Engineering*, 23(8):1244–1260.

[64] Deparis, É., Abel, M. H., and Mattioli, J. (2011). Modeling a social collaborative platform with standard ontologies. *Proceedings - 7th International Conference on Signal Image Technology and Internet-Based Systems, SITIS 2011*, pages 167–173.

[65] Dey, N., Ashour, A. S., Shi, F., Fong, S. J., Manuel, J., and Tavares, R. S. (2018). Medical cyber-physical systems: A survey. *Journal of Medical Systems*, 42(4):1–13.

[66] Dobson, S., Sterritt, R., Nixon, P., and Hinchey, M. (2010). Fulfilling The Vision of Autonomic Computing. *IEEE Computer*, 43(1):35–41.

[67] Dorri, A., Kanhere, S. S., and Jurdak, R. (2018). Multi-Agent Systems: A Survey. *IEEE Access*, 6:28573–28593.

[68] Dressler, F. (2018). Cyber Physical Social Systems: Towards Deeply Integrated Hybridized Systems. In *2018 International Conference on Computing, Networking and Communications, ICNC 2018*, pages 420–424. IEEE.

[69] Duan, F., Li, X., Liu, Y., and Fang, Y. (2011). Towards Autonomic Computing: A New Self-Management Method. In *International Conference on Artificial Intelligence and Computational Intelligence*, pages 292–299. Springer-Verlag.

[70] Edmonds, B. (1998). Modeling socially intelligent agents. *Applied Artificial Intelligence*, 12(7-8):677–699.

[71] Egrist (2019). GRiST Mental Health Decision Support for Everyone.

[72] El-korany, A. M. and Khatab, S. M. (2012). Ontology-based Social Recommender System. *IAES International Journal of Artificial Intelligence*, 1(3):127.

[73] Ellis, C., Gibbs, S., and Rein, G. (1991). Groupware: Some Issues and Experiences. *Communications of the ACM*, 34(1):39–58.

[74] Emilio, N., Ferreyra, D., and Sch¨awel, J. (2016). Self-disclosure in Social Media: An Opportunity for Self-Adaptive Systems. In *REFSQ Workshops*, volume 1564.

[75] Engeström, J. (2005). Why some social network services work and others don't-Or: the case for object-centered sociality.

[76] Epstein, B. (2018). Social Ontology. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University.

[77] Erétéo, G., Limpens, F., Gandon, F., Corby, O., Buffa, M., Leitzelman, M., and Sander, P. (2011). Semantic Social Network Analysis , a concrete case. *Knowledge Organization*, pages 122–156.

[78] Eze, B., Kuziemsky, C., Peyton, L., Middleton, G., and Mouttham, A. (2010). Policy-based data integration for e-health monitoring processes in a B2B environment: Experiences from Canada. *Journal of Theoretical and Applied Electronic Commerce Research*, 5(1):56–70.

[79] Eze, T. O., Anthony, R. J., Walshaw, C., and Soper, A. (2011). The Challenge of Validation for Autonomic and Self-Managing Systems. In *he Seventh International Conference on Autonomic and Autonomous Systems*, number 7, pages 128–133. IARIA.

[80] Farahani, A., Nazemi, E., Cabri, G., and Capodieci, N. (2017). Enabling autonomic computing support for the jade agent platform. *Scalable Computing*, 18(1):91–103.

[81] Farfan, F., Hristidis, V., Ranganathan, A., and Weiner, M. (2009). XOntoRank: Ontology-aware search of electronic medical records. In *2009 IEEE 25th International Conference on Data Engineering*, pages 820–831. IEEE.

[82] F.E., E. and E.L., T. (1960). *Socio-technical systems. Management sciences, models and Techniques*, volume II. Pergamon Press, Oxford.

[83] Ferreira, M. E. and Tereso, A. P. (2014). Software Tools for Project Management–Focus on Collaborative Management. *New Perspectives in Information Systems and Technologies*, 2:73–84.

[84] Finin, T., Ding, L., Zhou, L., and Joshi, A. (2005). Social networking on the semantic web. *The Learning Organization*, 12(5):418–435.

[85] Foster, D., McGregor, C., and El-Masri, S. (2005). A survey of agent-based intelligent decision support systems to support clinical management and research. *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 16–34.

[86] Franceschet, M., Gubiani, D., Montanari, A., and Piazza, C. (2009). From Entity Relationship to XML Schema: A Graph-Theoretic Approach. *Database and Xml Technologies, Proceedings*, 5679:165–179.

[87] Furrer, F. J. (2017). From ALGORITHMIC Computing to AUTONOMIC Computing. Technical report, Technische Universität Dresden.

[88] Gaál, Z., Szabó, L., Obermayer-Kovács, N., and Csepregi, A. (2015). Exploring the Role of Social Media in Knowledge Sharing. *The Electronic Journal of Knowledge Management*, 13(3):185–197.

[89] Ganek, A. G. and Corbi, T. A. (2003). The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1):5–18.

[90] Ghafoor, F. and Niazi, M. A. (2016). Using social network analysis of human aspects for online social network software: a design methodology. *Complex Adaptive Systems Modeling*, 4(1):14.

[91] Golbeck, J. and Rothstein, M. (2008). Linking Social Networks on the Web with FOAF: A Semantic Web Case Study. In *23rd national conference on Artificial intelligence*, volume 2, pages 1138–1143. AAAI Press.

[92] Golpayegani, F. (2015). Multi-agent collaboration in distributed self-adaptive systems. *Proceedings - 2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASOW 2015*, pages 146–151.

[93] González, E. J., Hamilton, A. F., Moreno, L., Marichal, R. L., and Muñoz, V. (2006). Software experience when using ontologies in a multi-agent system for automated planning and scheduling. *Software: Practice and Experience*, 36(7):667–688.

[94] Greif, I. (1988). *Computer-supported cooperative work: a book of readings*. Morgan Kaufmann Publishers Inc., San Francisco, USA.

[95] Griffiths, F., Cave, J., Boardman, F., Ren, J., Pawlikowska, T., Ball, R., Clarke, A., and Cohen, A. (2012). Social networks - The future for health care delivery. *Social Science and Medicine*, 75(12):2233–2241.

[96] Grimm, S., Hitzler, P., and Abecker, A. (2007). Knowledge Representation and Ontologies Logic, Ontologies and Semantic Web Languages. *Semantic Web Services*, pages 51–105.

[97] Gruber, T. R. and Others (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220.

[98] Gurgen, L., Gunalp, O., Benazzouz, Y., and Galissot, M. (2013). Self-aware Cyber-physical Systems and Applications in Smart Buildings and Cities. *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013*, pages 1149–1154.

[99] Gutiérrez, S. A. and Branch, J. W. (2011). A comparison between expert systems and autonomic computing plus mobile agent approaches for fault management . *Una comparación entre los enfoques basados en sistemas expertos y computación autónoma más agentes móviles para la gestión de fallas*, 78(168):173–180.

[100] Haake, J., Hussein, T., Joop, B., Lukosch, S., Veiel, D., and Ziegler, J. (2009). Technischer Bericht Nr . 2009-02 Context Modelling for Adaptive Collaboration. *Management*, 77251(60):1–49.

[101] Hamasaki, M. and Matsuo, Y. (2007). Ontology extraction using social network. ... *Workshop on Semantic ...*, 18700163(18700163).

[102] Haque, S. A., Aziz, S. M., and Rahman, M. (2014). Review of cyber-physical system in healthcare. *International Journal of Distributed Sensor Networks*, 2014.

[103] Hazeyama, A. (2014). Collaborative Software Engineering Learning Environment Associating Artifacts Management with Communication Support. In *The 3rd International Conference on Advanced Applied Informatics*, pages 592–596. IEEE.

[104] Holmquist, L. E., Mattern, F., Schiele, B., Alahuhta, P., Beigl, M., and Gellersen, H.-W. (2001). Smart-its friends: A technique for users to easily establish connections between smart artefacts. In *international conference on Ubiquitous Computing*, pages 116–122. Springer.

[105] Hong, I., Youn, H., Chun, I., and Lee, E. (2011). Autonomic Computing Framework for Cyber-Physical Systems. In *International Conference on Advances in Computing, Control, and Telecommunication Technologies*, volume 1, pages 140–143. ACEEE.

[106] Horn, P. (2001). Autonomic computing: IBM's Perspective on the State of Information Technology.

[107] Horridge, M. and Bechhofer, S. (2011). The OWLAPI: a Java API for OWL ontologies. *Semantic Web*, 2(1):11–21.

[108] Hristoskova, A., Sakkalis, V., Zacharioudakis, G., Tsiknakis, M., and De Turck, F. (2014). Ontology-Driven Monitoring of Patient's Vital Signs Enabling Personalized Medical Detection and Alert. *Sensors*, 14(1):1598–1628.

[109] Hu, H., Ahn, G.-J., and Kulkarni, K. (2011). Ontology-based Policy Anomaly Management for Autonomic Computing. In *7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, pages 487–494, Orlando. IEEE.

[110] Huebscher, M. C. and McCann, J. a. (2004). Evaluation Issues in Autonomic Computing. In *International Conference on Grid and Cooperative Computing*, pages 597–608. Springer-Verlag.

[111] Hussain, N. and Wang, H. H. (2014). Semantic Enabled Social-Collaborative Research Framework for Proteomics Domain. In *The 3rd International Conference on ASE BigData/SocialInformatics/PASSAT/BioMedCom*, pages 1–11, MA, Cambridge. ASE.

[112] Hussain, N., Wang, H. H., and Buckingham, C. (2018). Policy Based Generic Autonomic Adapter For A Context-Aware Social-Collaborative System. In *International Conference on Intelligent Systems and Computer Vision (ISCV)*, pages 1–9, Fez. IEEE.

[113] Hussain, N., Wang, H. H., and Buckingham, C. D. (2019). Artifact-Centric Semantic Social-Collaborative Network In An Online Healthcare Context. In *17th International Conference on e-Society*, Utrecht. IADIS Digital Library.

[114] Hussain, N., Wang, H. H., Buckingham, C. D., and Zhang, X. (2020). Software Agent-Centric Semantic Social Network for Cyber-Physical Interaction and Collaboration. *International Journal of Software Engineering and Knowledge Engineering*, 30(06):859–893.

[115] Ingeol, C., Park, J., Kim, W., Kang, W., Lee, H., and Park, S. (2010). Autonomic computing technologies for cyber-physical systems. In *Advanced Communication Technology (ICACT), 2010 The 12th International Conference on*, volume 2, pages 1009–1014.

[116] Irani, Z., Sharif, A. M., Papadopoulos, T., and Love, P. E. D. (2017). Social media and Web 2.0 for knowledge sharing in product design. *Production Planning & Control*, 28(13):1047–1065.

[117] Jian Quan OUYANG, Dian Xi SHI, Bo DING, J. F. and WANG, H. M. (2009). Policy Based Self-Adaptive Scheme in Pervasive Computing. *Wireless Sensor Network*, 1(1):1–60.

[118] Joly, A., Maret, P., and Daigremont, J. (2009). Context-awareness, the missing block of social networking. *International Journal of Computer Science and Applications*, 6(2):50–65.

[119] Kabir, M. A., Han, J., Yu, J., and Colman, A. (2014). User-centric social context information management: An ontology-based approach and platform. *Personal and Ubiquitous Computing*, 18(5):1061–1083.

[120] Kaldoudi, E., Dovrolis, N., Giordano, D., and Dietze, S. (2011). Educational Resources as Social Objects in Semantic Social Networks. In *Proceedings of the Linked Learning*, volume 717. CiteSeerX.

[121] Kamberov, R. (2016). Using social paradigms in smart cities mobile context-aware computing. In *11th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–5. IEEE.

[122] Kang, H., Li, X., and Moran, P. J. (2006). Autonomic Sensor Networks: A New Paradigm for Collaborative Information Processing. In *2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, pages 258–268. IEEE.

[123] Katayama, T., Wilkinson, M. D., Micklem, G., Kawashima, S., Yamaguchi, A., Nakao, M., Yamamoto, Y., Okamoto, S., Oouchida, K., Chun, H. W., Aerts, J., Afzal, H., Antezana, E., Arakawa, K., Aranda, B., Belleau, F., Bolleman, J., and Bonnal, R. J. (2013). The 3rd DBCLS BioHackathon: Improving life science data integration with Semantic Web technologies. *Journal of Biomedical Semantics*, 4(1):1.

[124] Kennedy, C. M. (2010). Decentralised metacognition in context-aware autonomic systems : Some key challenges. In *4th AAAI Conference on Metacognition for Robust Social Systems (AAAIWS'10-04)*, pages 34–41. ACM Digital Library.

[125] Kephart, J. O. and Walsh, W. E. (2004). An Artificial Intelligence Perspective on Autonomic Computing Policies. In *Fifth IEEE International Workshop on Policies for Distributed Systems and Networks, POLICY '04*, pages 3–12, NY. IEEE.

[126] Kershaw, A., Ruppel, K., and Chan, C. (2013). Oracle Social Network.

[127] Khorakhun, C. and Bhatti, S. (2013). Remote Health Monitoring Using Online Social Media Systems. *Saleem.Host.Cs.St-Andrews.Ac.Uk*, pages 23–25.

[128] King, T. M., Ramirez, A. E., Cruz, R., and Clarke, P. J. (2007). An Integrated Self-Testing Framework for Autonomic Computing Systems. *JOURNAL OF COMPUTERS*, 2(9):37–49.

[129] Klein, C., Schmid, R., Leuxner, C., Sitou, W., and Spanfelner, B. (2008). A survey of context adaptation in autonomic computing. *Proceedings - 4th International Conference on Autonomic and Autonomous Systems, ICAS 2008*, pages 106–111.

[130] Koch, M. (2008). CSCW and Enterprise 2.0 - towards an integrated perspective. In *21th Bled eConference eCollaboration: Overcoming Boundaries Through Multi-Channel Interaction*, pages 1–15, Bled.

[131] Kosonen, I. and Ma, X. (2016). Traffic Signal Control with Autonomic Features. *Autonomic Road Transport Support Systems, Autonomic Systems*, pages 253–267.

[132] Kożuch, B. and Sienkiewicz-Małyjurek, K. (2016). Factors of Effective Inter-Organizational Collaboration: a Framework for Public Management. *Transylvanian Review of Administrative Sciences*, 12(47):97–115.

[133] Krämer, B. and Conrad, J. (2017). Social Ontologies Online: The Representation of Social Structures on the Internet. *Social Media and Society*, 3(1):1–11.

[134] Kranz Matthias, L. R., Michahelles, F., Kranz, M., and Roalter, L. (2010). Things that twitter: social networks and the internet of things. In *Kranz, Matthias, Luis Roalter, and Florian Michahelles. "Things that twitter: social networks and the internet of things." In What can the Internet of Things do for the Citizen (CIoT) Workshop at The Eighth International Conference on Pervasive Computing*, pages 1–10.

[135] Kumazawa, T., Hara, K., Endo, A., and Taniguchi, M. (2017). Supporting collaboration in interdisciplinary research of water–energy–food nexus by means of ontology engineering. *Journal of Hydrology: Regional Studies*, 11:31–43.

[136] Kwiatkowski, T., Choi, E., Artzi, Y., and Zettlemoyer, L. (2013). Scaling Semantic Parsers with On-the-fly Ontology Matching. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, number October, pages 1545–1556. Association for Computational Linguistics.

[137] Lapointe, L., Ramaprasad, J., and Vedel, I. (2014). Creating health awareness: A social media enabled collaboration. *Health and Technology*, 4(1):43–57.

[138] Lasierra, N., Alesanco, A., Guillén, S., and Garc\'\ia, J. (2013). A three stage ontology-driven solution to provide personalized care to chronic patients at home. *Journal of biomedical informatics*, 46(3):516–529.

[139] Lasierra, N., Roldán, F., Alesanco, A., and García, J. (2014). Towards improving usage and management of supplies in healthcare: An ontology-based solution for sharing knowledge. *Expert Systems with Applications*, 41(14):6261–6273.

[140] Lee, I., Sokolsky, O., Chen, S., Hatcliff, J., Jee, E., Kim, B., King, A., Mullen-Fortino, M., Park, S., Roederer, A., and Venkatasubramanian, K. (2012). Computer Sciences Commons Invited Paper in Special Issue on Cyber-Physical Systems.Challenges and Research Directions in Medical Cyber-Physical Systems. *Proceedings of the IEEE*, 100(1):75–90.

[141] Lehtihet, E., Strassner, J., Agoulmine, N., and Foghlú, M. Ó. (2006). Ontology-Based Knowledge Representation for Self-governing Systems. pages 74–85.

[142] Leist, A. K. (2013). Social Media Use of Older Adults: A Mini-Review. *Gerontology*, 59(4):378–384.

[143] Lewis, D., Feeney, K., Carey, K., Tiropanis, T., and Courtenage, S. (2005). Semantic-based policy engineering for autonomic systems. *Lecture Notes in Computer Science*, 3457:152–164.

[144] Lima, C., Antunes, M., Gomes, D., Aguiar, R., and Mota, T. (2014). A Context-Aware Framework for Collaborative Activities in Pervasive Communities. *International Journal of Distributed Systems and Technologies*, 5(2):31–43.

[145] Lima, T., dos Santos, R. P., Oliveira, J., and Werner, C. (2016). The importance of socio-technical resources for software ecosystems management. *Journal of Innovation in Digital Ecosystems*, 3(2):98–113.

[146] Limpens, F., Gandon, F., and Buffa, M. (2008). Bridging ontologies and folksonomies to leverage knowledge sharing on the social web: A brief survey. *Aramis 2008 - 1st International Workshop on Automated engineeRing of Autonomous and runtiMe evolvIng Systems, and ASE2008 the 23rd IEEE/ACM Int. Conf. Automated Software Engineering*, pages 13–18.

[147] Lin, C.-H., Ho, P.-H., and Lin, H.-C. (2014). Framework for NFC-Based Intelligent Agents: A Context-Awareness Enabler for Social Internet of Things. *International Journal of Distributed Sensor Networks*, 10(2):1–16.

[148] Liptchinsky, V., Khazankin, R., Schulte, S., Satzger, B., Truong, H. L., and Dustdar, S. (2014). On modeling context-aware social collaboration processes. *Information Systems*, 43:66–82.

[149] Liu, H. and Parashar, M. (2003). DIOS++: A Framework for Rule-Based Autonomic Management of Distributed Scientific Applications. In *European Conference on Parallel Processing*, pages 66–73. Springer Berlin Heidelberg.

[150] Lobo, J., Bhatia, R., and Naqvi, S. (1999). A Policy Description Language. In *Sixteenth national conference on Artificial intelligence*, pages 291–298, CA. American Association for Artificial Intelligence.

[151] Lohmann, S., Dietzold, S., Heim, P., and Heino, N. (2009). A web platform for social requirements engineering. *Software Engineering*, pages 309–315.

[152] Loukil, F., Ghedira-Guegan, C., Benharkat, A. N., Boukadi, K., and Maamar, Z. (2017). Privacy-Aware in the IoT Applications: A Systematic Literature Review. *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, LNCS 10573(October):552–569.

[153] Lundberg, N. and Sandahl, T. I. (1999). Artifacts in work practice. *From Paper to Digital Documents*, page 123.

[154] Lundberg, N. and Sandahl, T. I. (2000). What do artifacts mean to us in work? In *22nd Information Systems Research Seminar in Scandinavia (IRIS 22)*, pages 363–372. Enterprise Architectures for Virtual Organizations.

[155] Ma, Z., Schultz, M. J., Christensen, K., Værbak, M., Demazeau, Y., and Jørgensen, B. N. (2019). The application of ontologies in multi-agent systems in the energy sector: A scoping review. *Energies*, 12(16):1–31.

[156] Maamar, Z., Buregio, V., and Sellami, M. (2015). Collaborative Enterprise Applications Based on Business and Social Artifacts. In *24th International Conference on Enabling Technologies: Infrastructures for Collaborative Enterprises*, pages 150–155. IEEE.

[157] Malik, K. S., Prakash, N., and Rizvi, S. (2011). Ontology Creation towards an Intelligent Web: Some Key Issues Revisited. *International Journal of Engineering and Technology*, 3(1).

[158] Marie, N. and Gandon, F. (2011). Social Objects Description and Recommendation in Multidimensional Social Networks: OCSO Ontology and Semantic Spreading Activation. *Proceedings - 2011 IEEE International Conference on Privacy, Security, Risk and Trust and IEEE International Conference on Social Computing, PASSAT/SocialCom 2011*, pages 1415–1420.

[159] Marsella, S. C., Pynadath, D. V., and Read, S. J. (2004). PsychSim: Agent-based modeling of social interactions and influence. In *Proceedings of the international conference on cognitive modeling*, volume 36, pages 243–248.

[160] Martin, A. (2015). Intelligent Search Engine to a Semantic Knowledge Retrieval in the Digital Repositories. *International Journal on Advances in Intelligent Systems*, 8(1):67–76.

[161] Mata, F. J. and Quesada, A. (2014). Web 2.0, Social Networks and E-commerce as Marketing Tools. *Journal of Theoretical and Applied Electronic Commerce Research*, 9(1):56–69.

[162] McCarthy, J. (2007). From here to human-level AI. *Artificial Intelligence*, 171(18):1174–1182.

[163] Merdan, M., Hoebert, T., List, E., and Lepuschitz, W. (2019). Knowledge-based cyber-physical systems for assembly automation. *Production and Manufacturing Research*, 7(1):223–254.

[164] Mezghani, E., Halima, R. B., Rodriguez, I. B., Mezghani, E., Halima, R. B., Rodriguez, I. B., Model, K. D. A., Mezghani, E., Halima, R. B., Rodriguez, I. B., and Drira, K. (2012). A Model Driven Approach for Automated Design of Context-Aware Autonomic Architectures.

[165] Miller, B. (2005). The autonomic computing edge: The role of the human in autonomic systems. Technical report.

[166] Miller, S. J. (2013). Introduction to Ontology Concepts and Terminology. pages 1–45.

[167] Mohammed, M., Khan, M. B., and Bashier, E. B. M. (2016). *Machine learning: algorithms and applications*. Crc Press.

[168] Mola, O. and Bauer, M. (2012). Policy-Based Autonomic Collaboration for Cloud Management. In *The Seventh International Multi-Conference on Computing in the Global Information Technology (ICCGI)*, pages 288–293.

[169] Mukhija, A. and Glinz, M. (2005). The CASA Approach to Autonomic Applications. In *5th IEEE Workshop on Applications and Services in Wireless Networks (ASWN 2005)*, number July, pages 173–182, Paris. IEEE.

[170] Mulcahy, J. J., Huang, S., and Mahgoub, I. (2015). Autonomic Computing and VANET. In *IEEE SoutheastCon*, pages 1–7, Florida. IEEE.

[171] Murthy, D. and Lewis, J. P. (2015). Social Media, Collaboration, and Scientific Organizations. *American Behavioral Scientist*, 59(1):149–171.

[172] Musumeci, F., Rottondi, C., Nag, A., Macaluso, I., Zibar, D., Ruffini, M., and Tornatore, M. (2018). An overview on application of machine learning techniques in optical networks. *IEEE Communications Surveys & Tutorials*, 21(2):1383–1408.

[173] Nan, B. L., Qiang, Y., and Jiachun, D. (2010). Mobile takes social computing beyond Web 2.0.

[174] Nansen, B., van Ryn, L., Vetere, F., Robertson, T., Brereton, M., and Douish, P. (2014). An Internet of Social Things. In *26th Australian Computer-Human Interaction Conference on Designing Futures: the Future of Design*, pages 87–96, Sydney. ACM Digital Library.

[175] Nazarenko, A. A. and Camarinha-Matos, L. M. (2017). Towards Collaborative Cyber-Physical Systems. In *International Young Engineers Forum, YEF-ECE 2017*, pages 12–17. IEEE.

[176] Nejdl, W., Olmedilla, D., Winslett, M., and Zhang, C. C. (2005). Ontology-Based Policy Specification and Management. *Computer*, 3532:290–302.

[177] Neumann, M., O'Murchu, I., Breslin, J., Decker, S., Hogan, D., and MacDonaill, C. (2005). Semantic social network portal for collaborative online communities. *Journal of European Industrial Training*, 29(6 SPEC. ISS.):472–487.

[178] Nimalasena, A. and Getov, V. (2015). Context-aware Approach for Determining the Threshold Price in Name-Your-Own-Price Channels. In *International Conference on Context-Aware Systems and Applications*, pages 83–93. Springer, Cham.

[179] Norman, C. (2012). Evaluation & Social Media.

[180] Obeid, C., Lahoud, I., El Khoury, H., and Champin, P.-A. (2018). Ontology-based Recommender System in Higher Education. *Companion of the The Web Conference 2018 on The Web Conference 2018*, 2:1031–1034.

[181] Oellinger, T. and pinar oezden Wennerberg (2006). Ontology Based Modeling and Visualization of Social Networks for the Web. In *INFORMATIK 2006–Informatik für Menschen–Band 2, Beiträge der 36*, pages 489–497.

[182] Oliveira, F. F., Antunes, J. C. P., and Guizzardi, R. S. S. (2007). Towards a Collaboration Ontology. In *2nd Workshop on Ontologies and Metamodeling in Software and Data Engineering*, pages 97–108.

[183] Oliveira, J., Souza, J. M. D., and Perazolo, M. (2006). Managing Knowledge about Resources for Autonomic Computing. In *1st latin american autonomic computing symposium*, pages 124–126.

[184] Omicini, A., Ricci, A., and Viroli, M. (2008). Artifacts in the A&A Meta-model for Multi-agent Systems. *Autonomous Agents and Multi-Agent Systems*, 17(3):432–456.

[185] Ongenae, F., De Backere, F., Steurbaut, K., Colpaert, K., Kerckhove, W., Decruyenaere, J., and De Turck, F. (2010). Towards computerizing intensive care sedation guidelines: Design of a rule-based architecture for automated execution of clinical guidelines. *BMC Medical Informatics and Decision Making*, 10(1).

[186] Osterrieder, A. (2013). The value and use of social media as communication tool in the plant sciences. *Plant Methods*, 9(26):1–6.

[187] Overgoor, G., Chica, M., Rand, W., and Weishampel, A. (2019). Letting the computers take over: using AI to solve marketing problems. *California Management Review*, 61(4):156–185.

[188] Paganelli, F. and Giuli, D. (2008). Context-aware information services to support tourist communities. *Information Technology & Tourism*, 10(4):313–327.

[189] Papamarkos, G., Poulovassilis, A., and Wood, P. T. (2003). Event-condition-action rule languages for the semantic web. *Workshop on Semantic Web*, pages 855–864.

[190] Park, J., Yoon, T., and Lee, S. (2015). Generating System Knowledge for Autonomic Control. *Indian Journal of Science and Technology*, 8(18):1–4.

[191] Passant, a. (2007). Using Ontologies to Strengthen Folksonomies and Enrich Information Retrieval in Weblogs. *International Conference on Weblogs and Social Media.*

[192] Pessoa, R., Calvi, C., and Others (2007). Semantic Context Reasoning Using Ontology Based Models. *3th open European summer school and IFIP TC6. 6 conference on Dependable and adaptable networks and services*, pages 44–51.

[193] Petrick, I. J. (2013). Networked Innovation: Using Roadmapping to Facilitate Coordination, Collaboration and Cooperation. In Moehrle, M. G., Isenmann, R., and Phaal, R., editors, *Technology Roadmapping for Strategy and Innovation*, pages 31–46, Berlin, Heidelberg. Springer-Verlag.

[194] Pham, D. T. and Afify, A. A. (2005). Machine-learning techniques and their applications in manufacturing. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 219(5):395–412.

[195] Rhayem, A., Ahmed Mhiri, M. B., Salah, M. B., and Gargouri, F. (2017). Ontology-based system for patient monitoring with connected objects. *Procedia Computer Science*, 112:683–692.

[196] Ricci, A., Viroli, M., and Piancastelli, G. (2011). SimpA: An agent-oriented approach for programming concurrent applications on top of Java. *Science of Computer Programming*, 76(1):37–62.

[197] Romano, P., Giugno, R., and Pulvirenti, A. (2011). Tools and collaborative environments for bioinformatics research. *Briefings in Bioinformatics*, 12(6):549–561.

[198] Romeikat, R., Bauer, B., Sanneck, H., and Schmelz, C. (2007). A Policy-Based System for Network-Wide Configuration Management. *18th WWRF - Wireless World Reasearch Forum.*

[199] Rufus, R., Nick, W., Shelton, J., and Esterline, A. (2016). An autonomic computing system based on a rule-based policy engine and Artificial Immune Systems. In *CEUR Workshop Proceedings*, volume 1584, pages 105–108.

[200] Russell, S. and Norvig, P. (2003). Artificial Intelligence: A Modern Approach. *Artificial Intelligence: A Modern Approach*, pages 111–114.

[201] Salehie, M. and Tahvildari, L. (2009). Self-Adaptive Software: Landscape and Research Challenges. *ACM Transactions on Autonomous and Adaptive Systems*, 4(2):1–42.

[202] Samaan, N. and Karmouch, A. (2009). Towards Autonomic Network Management: an Analysis of Current and Future Research Directions. *IEEE Communications Surveys & Tutorials*, 11(3):22–36.

[203] Sánchez, B. B., Alcarria, R., Sánchez-Picot, Á., and Sánchez-De-Rivera, D. (2017). A methodology for the design of application-specific Cyber-Physical Social Sensing co-simulators. *Sensors (Switzerland)*, 17(10):1–46.

[204] Schaeffer-Filho, A., Lupu, E., and Sloman, M. (2015). Federating Policy-Driven Autonomous Systems: Interaction Specification and Management Patterns. *Journal of Network and Systems Management*, 23(3):753–793.

[205] Schmidt, D., Bordini, R. H., Meneguzzi, F., and Vieira, R. (2015). An Ontology for Collaborative Tasks in Multi-agent Systems. *Ontobras*, 1442.

[206] Seleznyov, A. and Hailes, S. (2004). Distributed Knowledge Management for Autonomous Access Control in Computer Networks.

[207] Sen, S., Rahaman, Z., Crawford, C., and Yücel, O. (2018). Agents for Social (Media) Change. In *17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018)*, number Aamas, pages 1198–1202. ACM.

[208] Serpanos, D. (2018). The Cyber-Physical Systems Revolution. *Computer*, 51(3):70–73.

[209] Seydoux, N., Drira, K., Hernandez, N., and Monteil, T. (2016). Autonomy through knowledge: How IoT-O supports the management of a connected apartment. *CEUR Workshop Proceedings*, 1783:67–78.

[210] Shadbolt, N., Hall, W., and Barners-Lee, T. (2006). The Semantic Web Revisited. *IEEE Intelligent Systems*, 21(3):96–101.

[211] Shankar, C., Talwar, Y., Iyer, S., Chen, Y., Milojicic, D., and Campbell, R. (2006). Specification-Enhanced Policies for Automated Management of Changes in IT Systems. In *20th conference on Large Installation System Administration*, pages 1–17. ACM Digital Library.

[212] Sheth, A., Anantharam, P., and Henson, C. (2013). Physical-Cyber-Social Computing: An Early 21st Century Approach. *Cyber-Physical-Social Systems*, 28:78–82.

[213] Shuaib, H., Anthony, R., and Pelc, M. (2011). A Framework for Certifying Autonomic Computing Systems. In *The Seventh International Conference on Autonomic and Autonomous Systems*, number c, pages 122–127.

[214] Shuman, J. and Twombly, J. (2010). Collaborative Networks Are The Organization:. *Vikalpa: The Journal for Decision Makers*, 35(1):1–13.

[215] Sifalakis, M., Fry, M., and Hutchison, D. (2010). Event Detection and Correlation for Network Environments. *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, 28(1):60–69.

[216] Silva, R., Oliveira, J., and de Souza, J. (2010). Improving Collaborative Knowledge Flow Design on Social Networks through Autonomic Computing Systems Properties. In *First Workshop of the Brazilian Institute for Web Science Research*, Rio de Janeiro.

[217] Singer, L. and Schneider, K. (2012). Influencing the adoption of software engineering methods using social software. *34th International Conference on Software Engineering (ICSE)*, pages 1325–1328.

[218] Singh, R., Salam, a., and Iyer, L. (2003). Using Agents and {XML} for Knowledge Representation and Exchange: An Intelligent Distributed Decision Support Architecture {(IDDSA)}. *{AMCIS} 2003 Proceedings.*

[219] Skillen, K. L., Chen, L., Nugent, C. D., Donnelly, M. P., Burns, W., and Solheim, I. (2014). Ontological user modelling and semantic rule-based reasoning for personalisation of Help-On-Demand services in pervasive environments. *Future Generation Computer Systems*, 34:97–109.

[220] Sloman, M. and Lupu, E. (2002). Security and Management Policy Specification. *IEEE Network*, 16(2):10–19.

[221] Smailovic, V. and Podobnik, V. (2016). BeFriend: A Context-Aware Ad-hoc Social Networking Platform. *Automatika – Journal for Control, Measurement, Electronics, Computing and Communications*, 57(1):58–65.

[222] Smirnov, A., Kashevnik, A., and Ponomarev, A. (2015). Multi-level self-organization in cyber-physical-social systems: Smart home cleaning scenario. *Procedia CIRP*, 30:329–334.

[223] Smirnov, A. and Levashova, T. (2013). Towards Methodology for Design of Context-Aware Decision Support Systems based on Knowledge Fusion Patterns. *International Workshop on Information Logistics, Knowledge Supply and Ontologies in Information Systems.*

[224] Smirnov, A., Levashova, T., and Kashevnik, A. (2018). Ontology-Based Resource Interoperability in Socio-Cyber-Physical Systems Collaboration scenario. *IT in Industry*, 6(2):19–24.

[225] Soto, M. V., Balls-Berry, J. E., Bishop, S. G., Aase, L. A., Timimi, F. K., Montori, V. M., Patten, C. A., Valdez Soto, M., Balls-Berry, J. E., Bishop, S. G., Aase, L. A., Timimi, F. K., Montori, V. M., and Patten, C. A. (2016). Use of Web 2.0 Social Media Platforms to Promote Community-Engaged Research Dialogs: A Preliminary Program Evaluation. *JMIR research protocols*, 5(3):1–11.

[226] Spillers, F. and Loewus-deitch, D. (2003). Temporal attributes of shared artifacts in collaborative task environments. In *HCI2003: Workshop on the Temporal Aspects of Tasks*, pages 1–11, Bath.

[227] Stathis, K. (2010). Autonomic Computing with Self-governed Super-agents. In *Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshop, SASOW 2010*, pages 76–79. IEEE.

[228] Stenmark, M. and Malec, J. (2013). Knowledge-based industrial robotics. *Frontiers in Artificial Intelligence and Applications*, 257:265–274.

[229] Sterritt, R. (2005). Autonomic Computing. *Innovations in Systems and Software Engineering*, 1(1):79–88.

[230] Sterritt, R., Mulvenna, M., and Lawrynowicz, A. (2004). Dynamic and Contextualised Behavioural Knowledge in Autonomic Communications. In *Workshop on Autonomic Communication*, volume 3457, pages 217–228. Springer Berlin Heidelberg.

[231] Stoilova, K., Stoilov, T., and Nikolov, K. (2013). Autonomic properties in traffic control. *Cybernetics and Information Technologies*, 13(4):18–32.

[232] Stroka, S. (2008). Knowledge Representation Technologies in the Semantic Web. *World Wide Web Internet And Web Information Systems*, page 21.

[233] Tamma, V., Blacoe, I., Lithgow-Smith, B., and Wooldridge, M. (2005). Introducing autonomic behaviour in semantic web agents. In *International Semantic Web Conference*, pages 653–667. Springer Berlin Heidelberg.

[234] Tang, S., Yuan, J., Xufei Mao, Xiang-Yang Li, Chen, W., and Dai, G. (2011). Relationship classification in large scale online social networks and its impact on information propagation. In *Proceedings - IEEE INFOCOM*, pages 2291–2299.

[235] Tenorth, M., Bartels, G., and Beetz, M. (2014). Knowledge-based specification of robot motions. *Frontiers in Artificial Intelligence and Applications*, 263:873–878.

[236] Törngren, M. and Grogan, P. (2018). How to Deal with the Complexity of Future Cyber-Physical Systems? *Designs*, 2(4):1–16.

[237] Tretola, G. and Zimeo, E. (2009). Monitoring workflows execution using ECA rules. *ICSOFT 2009 - 4th International Conference on Software and Data Technologies, Proceedings*, 2.

[238] Tripathi, K. P. (2011). A Review on Knowledge-based Expert System : Concept and Architecture. *Artificial Intelligence Techniques - Novel Approaches & Practical Applications*, 4(4):19–23.

[239] Trsleff, S., Hildebrandt, C., Fay, A., Daun, M., and Brings, J. (2018). Developing Ontologies for the Collaboration of Cyber-Physical Systems: Requirements and Solution Approach. In *th International Workshop on Emerging Ideas and Trends in the Engineering of Cyber-Physical Systems*, pages 25–32. IEEE.

[240] Truszkowski, W. F., Hinchey, M. G., Rash, J. L., and Rouff, C. A. (2006). Autonomous and Autonomic Systems: A Paradigm for Future Space Exploration Missions. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 36(3):279–291.

[241] Trypuz, R. and Kuziński, D. (2016). Why use RDF / OWL rather than XML to represent and share global Legal Entity Identifiers (LEIs) and related LEI reference data. Technical report, MakoLab S.A. Semantic Web Research and Development Team.

[242] Vassev, E. and Hinchey, M. (2011). Knowledge representation and awareness in autonomic service-component ensembles - State of the art. *Proceedings - 2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops, ISORCW 2011*, pages 110–119.

[243] Ventola, C. L. (2014). Social media and health care professionals: benefits, risks, and best practices. *P & T : a peer-reviewed journal for formulary management*, 39(7):491–520.

[244] Verma, D. C. (2002). Simplifying network administration using policy-based management. *IEEE Network*, 16(2):20–26.

[245] Vollmer, T., Manic, M., and Linda, O. (2014). Autonomic intelligent cyber-sensor to support industrial control network awareness. *IEEE Transactions on Industrial Informatics*, 10(2):1647–1658.

[246] Von Der Weth, C., Abdul, A. M., and Kankanhalli, M. (2017). Cyber-Physical Social Networks. *ACM Transactions on Internet Technology*, 17(2).

[247] Wan, K. and Alagar, V. (2014). Achieving Dependability of Cyber Physical Systems with Autonomic Covering. In *IEEE 12th International Conference on Dependable, Autonomic and Secure Computing Achieving*, pages 139–145, Dalian. IEEE.

[248] Wang, F.-y., Zeng, D., Carley, K. M., and Mao, W. (2007). Social Computing : From Social Informatics. *IEEE Intelligent Systems*, 22(2):79–83.

[249] Warnier, M., Sinderen, M. V., and Brazier, F. M. T. (2010). Adaptive Knowledge Representation for a Self-Managing Home Energy Usage System. In *Fourth International Workshop on Enterprise Systems and Technology (I-WEST)*, pages 132–141, Athens. Scite Press.

[250] Wei, D., Ning, H., Qian, Y., and Zhu, T. (2018). Social relationship for physical objects. *International Journal of Distributed Sensor Networks*, 14(1).

[251] White, S. R., Hanson, J. E., Whalley, I., Chess, D. M., and Kephart, J. O. (2004). An Architectural Approach to Autonomic Computing. In *International Conference on Autonomic Computing (ICAC'04)*, pages 2–9, New York. IEEE.

[252] Wilson, C., Boe, B., Sala, A., Puttaswamy, K. P., and Zhao, B. Y. (2009). User interactions in social networks and their implications. *Proceedings of the fourth ACM european conference on Computer systems - EuroSys '09*, page 205.

[253] Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. Wiley Publishing, 2nd edition.

[254] Wooldridge, M. J. and Jennings, N. R. (1995). Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(2):115–152.

[255] XACML-V3.0 (2017). eXtensible Access Control Markup Language (XACML) Version 3.0.

[256] Xia, F. and Ma, J. (2011). Building Smart Communities with Cyber-Physical Systems. In *1st International Symposium on from Digital Footprints to Social and Community Intelligence*, pages 1–5. ACM.

[257] Xiao, Y. (2005). Artifacts and collaborative work in healthcare: Methodological, theoretical, and technological implications of the tangible. *Journal of Biomedical Informatics*, 38(1):26–33.

[258] Yagüe, M. I. (2006). Survey on XML-Based Policy Languages for Open Environments. *Journal of Information Assurance and Security*, 1(1):11–20.

[259] Yilma, B. A., Panetto, H., and Naudet, Y. (2019). A Meta-Model of Cyber-Physical-Social System: The CPSS Paradigm to Support Human-Machine Collaboration in Industry 4.0. In *20th IFIP WG 5.5 Working Conference on Virtual Enterprises*, pages 11–20. Springer, Cham.

[260] Zaher, N. A. and Buckingham, C. D. (2016). Moderating the Influence of Current Intention to Improve Suicide Risk Prediction. *AMIA Annu Symp Proc*, 2016:1274–1282.

[261] Zanni, A. (2015). Cyber-physical systems and smart cities Learn how smart devices , sensors , and actuators are advancing Internet of Things implementations. *IBM developerWorks*, (April):1–8.

[262] Zhang, C., Xu, X., and Chen, H. (2019). Theoretical foundations and applications of cyber-physical systems: a literature review. *Library Hi Tech*.

[263] Zhang, X., Zhao, Y., and Liu, W. (2015). A Method for Mapping Sensor Data to SSN Ontology. *International Journal of u-and e-Service*, 8(6):303–316.

[264] Zhao, Z., Schiller, E., Kalogeiton, E., Braun, T., Stiller, B., Garip, M. T., Joy, J., Gerla, M., Akhtar, N., and Matta, I. (2017). Autonomic communications in software-driven networks. *IEEE Journal on Selected Areas in Communications*, 35(11):2431–2445.

[265] Zhenping, L., Savkli, C., and Jones, L. (2006). Autonomic computing for spacecraft ground systems. In *2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06)*, pages 1–22. IEEE.

[266] Zhou, N., Delaval, G., Robu, B., Rutten, E., and Mehaut, J. F. (2016). Autonomic parallelism and thread mapping control on software transactional memory. In *IEEE International Conference on Autonomic Computing (ICAC)*, pages 189–198, Wurzburg. IEEE.

[267] Zhou, P., Zou, D., Hou, K.-M., and Zhang, Z. (2017). A Decentralized Compositional Framework for Dependable Decision Process in Self-Managed Cyber Physical Systems. *Sensor*, 17(11):1–33.

APPENDIX A

AAs Policy Configuration for GRiST Socio-Healthcare System

This section presents the full configuration of the policies for AA1, AA2, AA3 and Global RIE as discussed in the Chapter 5, Section (5.5.2.1).

A.1 AA1 Policy Script

The full AA1 policy script for GRiST configured in XML format, which is shown below:

```
<AAPolicy aaID="aa1" managedResource="pulseMonitor" policyType="pulse
monitor␣policy">
<Rule ruleID="r1">
<Event eventID="e1" eventSource="pulseMonitor" event="pulseChange"/>
<Condition>
<Expression exprID="ex1" varName="heartRate" op="&lt;" value="40"/>
<Expression exprID="ex2" varName="heartRate" op="&gt;" value="140"/>
<Action actionID="a1" actionName="send-alert">
<Method name="alert">
<Parameter paramName="patientNode" value="10"/>
<Parameter paramName="message" value="sits␣down␣and␣breathes␣slowly␣and␣
    steadily␣until␣the␣pulse␣normal"/>
</Method>
</Action>
<Action actionID="a2" actionName="send-notification">
<Method name="notify">
<Parameter paramName="ManagedResource" value="activitySensor"/>
<Parameter paramName="Event" value="e1"/>
</Method>
</Action>
</Condition>
</Rule>
</AAPolicy>
```

Listing A.1 : AA1 policy script in XML format.

A.2 AA2 Policy Script

The full AA2 policy script for GRiST configured in XML format, which is shown below:

```
<AAPolicy aaID="aa2" managedResource="activitySensor" policyType="activity␣
    sensor␣policy">
<Rule ruleID="r2">
<Event eventID="e2" eventSource="activitySensor" event="getNotified"/>
<Condition>
<Expression varName="physicalActivity" op="&lt;" value="1.4"/>
<Action actionID="a3" actionName="send-alert">
<Method name="alert">
<Parameter paramName="patientNode" value="10"/>
<Parameter paramName="message" value="do␣a␣manual␣assessment"/>
</Method>
</Action>
<Action actionID="a4" actionName="send-alert">
<Method name="alert">
<Parameter paramName="doctorNode" value="3"/>
<Parameter paramName="message" value="possible␣anxiety␣attack"/>
</Method>
</Action>
</Condition>
</Rule>
</AAPolicy>
```

Listing A.2 : AA2 policy script in XML format.

A.3 AA3 Policy Script

The full AA3 policy script for GRiST configured in XML format, which is shown below:

```
<AAPolicy aaID="aa3" managedResource="assessmentMonitor"
 policyType="assessment␣monitor␣policy">
  <Rule ruleID="r3">
    <Event eventID="e3" eventSource="assessmentMonitor" event="
        assessmentSubmitted" />
    <Condition>
      <Expression varName="riskAssessment" op="==" value="normal" />
      <Action actionID="a5" actionName="send-notification">
        <Method name="notify">
          <Parameter paramName="doctorNode" value="3" />
          <Parameter paramName="message" value="confirmed␣anxiety␣attack" />
        </Method>
      </Action>
      <Action actionID="a6" actionName="send-notification">
        <Method name="notify">
          <Parameter paramName="patientNode" value="10" />
          <Parameter paramName="message" value="confirmed␣anxiety␣attack" />
        </Method>
      </Action>
      <Action actionID="a7" actionName="send-notification">
        <Method name="notify">
```

```
          <Parameter paramName="familyNode" value="5" />
          <Parameter paramName="message" value="confirmed␣anxiety␣attack" />
        </Method>
      </Action>
    </Condition>
  </Rule>

  <Rule ruleID="r4">
    <Event eventID="e3" eventSource="assessmentMonitor" event="
        assessmentSubmitted" />
    <Condition>
      <Expression varName="riskAssessment" op="==" value="high" />
      <Action actionID="a8" actionName="send-alert">
        <Method name="alert">
          <Parameter paramName="carerNode" value="2" />
          <Parameter paramName="message" value="patient's␣need␣urgent␣care" />
        </Method>
      </Action>
      <Action actionID="a9" actionName="send-alert">
        <Method name="alert">
          <Parameter paramName="patientNode" value="10" />
          <Parameter paramName="message" value="you␣are␣at␣risk,␣emergency␣
              service␣is␣on␣the␣way" />
        </Method>
      </Action>
      <Action actionID="a10" actionName="send-alert">
        <Method name="alert">
          <Parameter paramName="familyNode" value="5" />
          <Parameter paramName="message" value="immediately␣give␣first␣aid" />
        </Method>
      </Action>
    </Condition>
  </Rule>
</AAPolicy>
```

Listing A.3 : AA3 policy script in XML format.

A.4 Global Policy Script

The full Global policy script for GRiST configured in XML format, which is shown below:

```
<GlobalPolicy>
  <InputAction>
    <Action id="a1" ref_aa="aa1" />
    <Action id="a2" ref_aa="aa1" />
    <Action id="a3" ref_aa="aa2" />
    <Action id="a4" ref_aa="aa2" />
    <Action id="a5" ref_aa="aa2" />
    <Action id="a6" ref_aa="aa2" />
    <Action id="a7" ref_aa="aa2" />
    <Action id="a8" ref_aa="aa2" />
    <Action id="a9" ref_aa="aa2" />
    <Action id="a10" ref_aa="aa2" />
  </InputAction>
  <RuleSet>
    <Rule ruleID="r5">
      <Condition>
        <Expression exprID="ex1" varName="heartRate" op="==" value="abnormal"
            />
          <Dispatch target="aa1">
            <Action> a1 </Action>
            <Action> a2 </Action>
          </Dispatch>
      </Condition>
    </Rule>
    <Rule ruleID="r6">
      <Condition>
        <Expression exprID="ex1" varName="heartRate" op="==" value="abnormal"
            />
        <Expression exprID="ex2" varName="physicalActivity" op="==" value="
            high" />
         <Dispatch target="aa2">
           <Action> a3 </Action>
           <Action> a4 </Action>
         </Dispatch>
      </Condition>
    </Rule>
    <Rule ruleID="r7">
      <Condition>
        <Expression exprID="ex1" varName="heartRate" op="==" value="abnormal"
            />
        <Expression exprID="ex2" varName="physicalActivity" op="==" value="low
            " />
        <Expression exprID="ex2" varName="riskAssessment" op="==" value="high"
             />
         <Dispatch target="aa3">
           <Action> a8 </Action>
           <Action> a9 </Action>
           <Action> a10 </Action>
         </Dispatch>
       </Condition>
    </Rule>
    <Rule ruleID="r8">
      <Condition>
        <Expression exprID="ex1" varName="heartRate" op="==" value="abnormal"
            />
        <Expression exprID="ex2" varName="physicalActivity" op="==" value="
            high" />
        <Expression exprID="ex2" varName="riskAssessment" op="==" value="
            normal" />
         <Dispatch target="aa3">
           <Action> a5 </Action>
           <Action> a6 </Action>
           <Action> a7 </Action>
         </Dispatch>
       </Condition>
     </Rule>
   </RuleSet>
```

```
</GlobalPolicy>
```

Listing A.4 : Global policy script in XML format.

APPENDIX B

Policy Specification and Configuration of eGRiST CDSS

This section presents the complete configuration of PSD language, policy configuration for AA1, AA2, AA3 and Global RIE that are integrated into eGRiST system as discussed in Chapter 6, Section (6.2.3) and Section (6.4.2) respectively.

B.1 Policy Schema Definition

The complete and workable PSD language is shown below:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
attributeFormDefault="unqualified">

<xs:element name="PolicySet">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="AAPolicy" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="GlobalPolicy"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="AAPolicy">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ManagedResourceList" minOccurs="0"/>
      <xs:element ref="SocialNetwork" minOccurs="0"/>
      <xs:element ref="AA"/>
    </xs:sequence>
   <xs:attribute type="xs:string" name="policyID" use="required"/>
   <xs:attribute type="xs:string" name="policyType" use="required"/>
   <xs:attribute type="xs:string" name="policyDescription" use="optional"/>
   <xs:attribute type="xs:string" name="ref_aa" use="required"/>
  </xs:complexType>
```

```
</xs:element>

<xs:element name="ManagedResourceList">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ManagedResource" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="ManagedResource">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="TriggerEvent"/>
    </xs:sequence>
    <xs:attribute type="xs:string" name="resourceID" use="required"/>
    <xs:attribute type="xs:string" name="resourceType" use="required"/>
    <xs:attribute type="xs:string" name="ref_sensor" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="TriggerEvent">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="and" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="or" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="not" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Event">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute type="xs:string" name="inputEvent" use="required"/>
        <xs:attribute type="xs:string" name="op" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name="SocialNetwork">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="NetworkResourceList"/>
    </xs:sequence>
   <xs:attribute type="xs:string" name="networkID" use="required"/>
  <xs:attribute type="xs:string" name="networkName" use="required"/>
 </xs:complexType>
</xs:element>

<xs:element name="NetworkResourceList">
  <xs:complexType>
    <xs:sequence>
```

```
      <xs:element ref="NetworkResource"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="NetworkResource">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="TriggerEvent"/>
    </xs:sequence>
    <xs:attribute type="xs:string" name="resourceID" use="required"/>
    <xs:attribute type="xs:string" name="resourceType" use="required"/>
    <xs:attribute type="xs:string" name="ref_sensor" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="AA">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element ref="SensorList" minOccurs="0"/>
      <xs:element ref="Monitor" minOccurs="0"/>
      <xs:element ref="LocalRIE" minOccurs="0"/>
      <xs:element ref="Executor" minOccurs="0"/>
    </xs:sequence>
   <xs:attribute type="xs:string" name="adapterID" use="required"/>
   <xs:attribute type="xs:string" name="adapterType" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="SensorList">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Sensor" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Sensor">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute type="xs:string" name="sensorID" use="required"/>
        <xs:attribute type="xs:string" name="sensorType" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name="Monitor">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="GetSensorData" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="Condition"/>
    </xs:sequence>
   <xs:attribute type="xs:string" name="monitorID" use="required"/>
```

```
    </xs:complexType>
  </xs:element>

  <xs:element name="GetSensorData">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:string" name="ref_sensor" use="required"/>
          <xs:attribute type="xs:string" name="ref_event" use="required"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="LocalRIE">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="GetMonitorData"/>
        <xs:element ref="Condition"/>
        <xs:element ref="TriggerAction"/>
      </xs:sequence>
     <xs:attribute type="xs:string" name="rieID" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="GetMonitorData">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="GetContextVariable" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
     <xs:attribute type="xs:string" name="ref_monitor" use="required"/>
     <xs:attribute type="xs:string" name="ref_condition" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="GetContextVariable">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:string" name="contextID" use="required"/>
          <xs:attribute type="xs:string" name="ref_expression" use="required"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="TriggerAction">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="While" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
     <xs:attribute type="xs:string" name="ref_condition" use="required"/>
    </xs:complexType>
  </xs:element>
```

```
<xs:element name="While">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ActionTemplate"/>
    </xs:sequence>
   <xs:attribute type="xs:string" name="condition" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="ActionTemplate">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute type="xs:string" name="actionID" use="required"/>
        <xs:attribute type="xs:string" name="actionType" use="required"/>
        <xs:attribute type="xs:string" name="category" use="optional"/>
        <xs:attribute type="xs:string" name="target_resource" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name="Executor">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="GetNotification"/>
    </xs:sequence>
   <xs:attribute type="xs:string" name="executorID" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="GetNotification">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Execute" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute type="xs:string" name="message" use="required"/>
    <xs:attribute type="xs:string" name="ref_sender" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="Execute">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Function"/>
    </xs:sequence>
   <xs:attribute type="xs:string" name="action" use="required"/>
   <xs:attribute type="xs:string" name="ref_command" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="Function">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Parameter" maxOccurs="unbounded" minOccurs="0"/>
```

```
        </xs:sequence>
      <xs:attribute type="xs:string" name="functionID" use="required"/>
    <xs:attribute type="xs:string" name="functionName" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="Parameter">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute type="xs:string" name="paramName" use="required"/>
        <xs:attribute type="xs:string" name="datavalue" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name="GlobalPolicy">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="GlobalRIE"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="GlobalRIE">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ActionList"/>
      <xs:element ref="Condition"/>
      <xs:element ref="Decision" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
   <xs:attribute type="xs:string" name="rieID" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="ActionList">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="GetAction" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="GetAction">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="and" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="or" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="not" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
   <xs:attribute type="xs:string" name="ref_aa" use="required"/>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="Action">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute type="xs:string" name="inputActionID" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name="Decision">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Command" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
   <xs:attribute type="xs:string" name="ref_condition" use="required"/>
   <xs:attribute type="xs:string" name="ref_expression" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="Command">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute type="xs:string" name="commandID" use="required"/>
        <xs:attribute type="xs:string" name="message" use="required"/>
        <xs:attribute type="xs:string" name="target_adapter" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name="Condition">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Expression" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
   <xs:attribute type="xs:string" name="conditionID" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="Expression">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="If" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="and" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="or" minOccurs="0"/>
      <xs:element ref="not" minOccurs="0"/>
    </xs:sequence>
   <xs:attribute type="xs:string" name="exprID" use="required"/>
   <xs:attribute type="xs:string" name="ref_event" use="optional"/>
  </xs:complexType>
</xs:element>

<xs:element name="If">
```

```
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:string" name="varName" use="required"/>
          <xs:attribute type="xs:string" name="op" use="required"/>
          <xs:attribute type="xs:string" name="datavalue" use="optional"/>
          <xs:attribute type="xs:string" name="ref_context" use="optional"/>
          <xs:attribute type="xs:string" name="ref_input_action" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="and">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="If" minOccurs="0"/>
        <xs:element ref="Event" minOccurs="0"/>
        <xs:element ref="Action" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="or">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="If" minOccurs="0"/>
        <xs:element ref="Event" minOccurs="0"/>
        <xs:element ref="Action" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="not">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="If" minOccurs="0"/>
        <xs:element ref="Event" minOccurs="0"/>
        <xs:element ref="Action" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  </xs:schema>
```

B.2 AAs Policy Configuration

The full configuration of AA1, AA2 and AA3 policy and Global RIE policy is shown below:

```
<PolicySet xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.aapolicyschema.com/policy.xsd">
  <!-- Configure AA1 policy with event -->
   <AAPolicy policyID="p1" policyType="depression" policyDescription="carer’s medium help intervention"
```

```
ref_aa="aa1">
<ManagedResourceList>
  <ManagedResource resourceID="mr1" resourceType="depression-status" ref_sensor="s1">
    <TriggerEvent>
      <or><Event inputEvent="depression-status" op="="> first-diagnosis </Event></or>
      <or><Event inputEvent="depression-status" op="="> recovery-single-episode </Event></or>
      <or><Event inputEvent="depression-status" op="="> recovery-repeat-episode </Event></or>
      <or><Event inputEvent="depression-status" op="="> relapse </Event></or>
      <or><Event inputEvent="depression-status" op="="> dont-know </Event></or>
    </TriggerEvent>
  </ManagedResource>
<ManagedResource resourceID="mr2" resourceType="act-on-voices" ref_sensor="s2">
  <TriggerEvent>
    <or><Event inputEvent="act-on-voices" op="="> 0.0 </Event></or>
    <or><Event inputEvent="act-on-voices" op="="> 0.1 </Event></or>
    <or><Event inputEvent="act-on-voices" op="="> 0.2 </Event></or>
    <or><Event inputEvent="act-on-voices" op="="> 0.3 </Event></or>
    <or><Event inputEvent="act-on-voices" op="="> 0.4 </Event></or>
    <or><Event inputEvent="act-on-voices" op="="> 0.5 </Event></or>
    <or><Event inputEvent="act-on-voices" op="="> 0.6 </Event></or>
    <or><Event inputEvent="act-on-voices" op="="> 0.7 </Event></or>
    <or><Event inputEvent="act-on-voices" op="="> 0.8 </Event></or>
    <or><Event inputEvent="act-on-voices" op="="> 0.9 </Event></or>
    <or><Event inputEvent="act-on-voices" op="="> 1.0 </Event></or>
  </TriggerEvent>
</ManagedResource>
<ManagedResource resourceID="mr3" resourceType="voice-urge-harm-self" ref_sensor="s3">
  <TriggerEvent>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 0.0 </Event></or>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 0.1 </Event></or>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 0.2 </Event></or>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 0.3 </Event></or>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 0.4 </Event></or>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 0.5 </Event></or>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 0.6 </Event></or>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 0.7 </Event></or>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 0.8 </Event></or>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 0.9 </Event></or>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 1.0 </Event></or>
  </TriggerEvent>
</ManagedResource>
<ManagedResource resourceID="mr4" resourceType="voice-urge-harm-other" ref_sensor="s4">
  <TriggerEvent>
    <or><Event inputEvent="voice-urge-harm-other" op="="> 0.0 </Event></or>
    <or><Event inputEvent="voice-urge-harm-other" op="="> 0.1 </Event></or>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 0.2 </Event></or>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 0.3 </Event></or>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 0.4 </Event></or>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 0.5 </Event></or>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 0.6 </Event></or>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 0.7 </Event></or>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 0.8 </Event></or>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 0.9 </Event></or>
    <or><Event inputEvent="voice-urge-harm-self" op="="> 1.0 </Event></or>
  </TriggerEvent>
```

```
</ManagedResource>
<ManagedResource resourceID="mr5" resourceType="delusions" ref_sensor="s5">
  <TriggerEvent>
    <or><Event inputEvent="delusions" op="="> 0.0 </Event></or>
    <or><Event inputEvent="delusions" op="="> 0.1 </Event></or>
    <or><Event inputEvent="delusions" op="="> 0.2 </Event></or>
    <or><Event inputEvent="delusions" op="="> 0.3 </Event></or>
    <or><Event inputEvent="delusions" op="="> 0.4 </Event></or>
    <or><Event inputEvent="delusions" op="="> 0.5 </Event></or>
    <or><Event inputEvent="delusions" op="="> 0.6 </Event></or>
    <or><Event inputEvent="delusions" op="="> 0.7 </Event></or>
    <or><Event inputEvent="delusions" op="="> 0.8 </Event></or>
    <or><Event inputEvent="delusions" op="="> 0.9 </Event></or>
    <or><Event inputEvent="delusions" op="="> 1.0 </Event></or>
   </TriggerEvent>
  </ManagedResource>
</ManagedResourceList>

<!-- Social network entities may also generate event if action sent to the network -->
<SocialNetwork networkID="network1" networkName="grace-care-network">
  <NetworkResourceList>
    <NetworkResource resourceID="" resourceType="" ref_sensor="">
      <TriggerEvent>
        <or><Event inputEvent="" op="="></Event></or>
      </TriggerEvent>
    </NetworkResource>
  </NetworkResourceList>
</SocialNetwork>

<!-- Configure AA1 components -->
<AA adapterID="aa1" adapterType="depression-concept-adapter">
  <SensorList>
    <Sensor sensorID="s1" sensorType="depression-status-sensor" />
    <Sensor sensorID="s2" sensorType="act-on-voices-sensor" />
    <Sensor sensorID="s3" sensorType="voice-urge-harm-self-sensor" />
    <Sensor sensorID="s4" sensorType="voice-urge-harm-others-sensor" />
    <Sensor sensorID="s5" sensorType="delusions-sensor" />
  </SensorList>

<!-- Configure AA1 monitor policy -->
<Monitor monitorID="m1">
  <GetSensorData ref_sensor="s1" ref_event="depression-status" />
  <GetSensorData ref_sensor="s2" ref_event="act-on-voices" />
  <GetSensorData ref_sensor="s3" ref_event="voice-urge-harm-self-sensor" />
  <GetSensorData ref_sensor="s4" ref_event="voice-urge-harm-self-others" />
  <GetSensorData ref_sensor="s5" ref_event="delusions" />

  <Condition conditionID="c1">
    <Expression exprID="expr1" ref_event="depression-status">
      <If varName="depression-status" op="==" datavalue="first-diagnosis"> low </If>
      <If varName="depression-status" op="==" datavalue="single-episode"> medium </If>
      <If varName="depression-status" op="==" datavalue="repeat-episode"> high </If>
      <If varName="depression-status" op="==" datavalue="relapse"> critical </If>
      <If varName="depression-status" op="==" datavalue="dont-know"> unknown </If>
    </Expression>
```

```
      <Expression exprID="expr2" ref_event="act-on-voices">
        <If varName="act-on-voices" op="&gt;=" datavalue="0.5"> likely </If>
        <If varName="act-on-voices" op="&lt;" datavalue="0.5"> unlikely </If>
        <If varName="act-on-voices" op="==;" datavalue="dont-know"> unknown </If>
      </Expression>
      <Expression exprID="expr3" ref_event="voice-urge-harm-self">
        <If varName="voice-urge-harm-self" op="&gt;=" datavalue="0.5"> likely </If>
        <If varName="voice-urge-harm-self" op="&lt;" datavalue="0.5"> unlikely </If>
        <If varName="voice-urge-harm-self" op="==" datavalue="dont-know"> unknown </If>
      </Expression>
      <Expression exprID="expr4" ref_event="voice-urge-harm-others">
        <If varName="voice-urge-harm-others" op="&gt;=" datavalue="0.5"> likely </If>
        <If varName="voice-urge-harm-others" op="&lt;" datavalue="0.5"> unlikely </If>
        <If varName="voice-urge-harm-others" op="==" datavalue="dont-know"> unknown </If>
      </Expression>
      <Expression exprID="expr5" ref_event="delusions">
        <If varName="delusions" op="&gt;=" datavalue="0.5"> likely </If>
        <If varName="delusions" op="&lt;" datavalue="0.5"> unlikely </If>
        <If varName="delusions" op="==" datavalue="dont-know"> unknown </If>
      </Expression>
    </Condition>
  </Monitor>

  <!-- Configure AA1 local rie policy -->
  <LocalRIE rieID="aa-rie1">
    <GetMonitorData ref_monitor="m1" ref_condition="c1">
      <GetContextVariable contextID="context1" ref_expression="expr1"/>
      <GetContextVariable contextID="context2" ref_expression="expr2"/>
      <GetContextVariable contextID="context3" ref_expression="expr3"/>
      <GetContextVariable contextID="context4" ref_expression="expr4"/>
      <GetContextVariable contextID="context5" ref_expression="expr5"/>
    </GetMonitorData>
    <Condition conditionID="c2">
      <Expression exprID="expr1">
        <and><If varName="depression-status" op="==" datavalue="low" ref_context="context1"> true </If></and>
        <and><If varName="act-on-voices" op="==" datavalue="unlikely" ref_context="context2"> true </If></and>
        <and><If varName="voice-urge-harm-self" op="==" datavalue="unlikely" ref_context="context3"> true </If></and>
        <and><If varName="voice-urge-harm-others" op="==" datavalue="unlikely" ref_context="context4"> true </If></and
        <and><If varName="delusions" op="==" datavalue="unlikely" ref_context="context5"> true </If></and>
      </Expression>
      <Expression exprID="expr2">
        <and><If varName="depression-stage" op="==" datavalue="medium" ref_context="context1"> true </If></and>
        <and><If varName="delusions-stage" op="==" datavalue="unlikely" ref_context="context2"> true </If></and>
        <and><If varName="hallucinations-stage" op="==" datavalue="unlikely" ref_context="context3"> true </If></and>
        <and><If varName="voice-urge-harm-others" op="==" datavalue="unlikely" ref_context="context4"> true </If></and
        <and><If varName="delusions" op="==" datavalue="unlikely" ref_context="context5"> true </If></and>
      </Expression>
      <Expression exprID="expr3">
        <and><If varName="depression-stage" op="==" datavalue="high" ref_context="context1"> true </If></and>
        <and><If varName="delusions-stage" op="==" datavalue="likely" ref_context="context2" > true </If></and>
        <and><If varName="hallucinations-stage" op="==" datavalue="likely" ref_context="context3"> true </If></and>
        <and><If varName="voice-urge-harm-others" op="==" datavalue="likely" ref_context="context4"> true </If></and>
        <and><If varName="delusions" op="==" datavalue="likely" ref_context="context5"> true </If></and>
      </Expression>
      <Expression exprID="expr4">
```

```
        <and><If varName="depression-stage" op="==" datavalue="critical" ref_context="context1"> true </If></and>
        <and><If varName="delusions-stage" op="==" datavalue="likely" ref_context="context2" > true </If></and>
        <and><If varName="hallucinations-stage" op="==" datavalue="likely" ref_context="context3"> true </If></and>
        <and><If varName="voice-urge-harm-others" op="==" datavalue="likely" ref_context="context4"> true </If></and>
        <and><If varName="delusions" op="==" datavalue="likely" ref_context="context5"> true </If></and>
      </Expression>
    </Condition>

    <TriggerAction ref_condition="c2">
      <While condition="expr1 = 'true'">
        <ActionTemplate actionID="a1" actionType="open-url" category="none" target_resource="resource name">
           galassify-action-open-url-template </ActionTemplate>
      </While>
      <While condition="expr2 = 'true'">
        <ActionTemplate actionID="a2" actionType="send-email" category="none" target_resource="resource name">
          galassify-action-send-email-template </ActionTemplate>
      </While>
      <While condition="(expr3 = 'true') and (expr4 = 'true')">
        <ActionTemplate actionID="a3" actionType="dial-emergency" category="none" target_resource="resource name">
          notify-emergency-template </ActionTemplate>
      </While>
    </TriggerAction>
  </LocalRIE>

  <!-- Configure AA1 executor rule -->
  <Executor executorID="exe1">
    <GetNotification message="yes" ref_sender="globalRIE" >
      <Execute action="action = 'a1'" ref_command="cmd1">
        <Function functionID="f1" functionName="sendEmail">
          <Parameter paramName="id" datavalue="1" />
          <Parameter paramName="name" datavalue="john" />
          <Parameter paramName="recepient" datavalue="abc@email.com" />
          <Parameter paramName="subject" datavalue="a subject line" />
          <Parameter paramName="textMsg" datavalue="We recommended you to seek help..." />
          <Parameter paramName="population_restriction" datavalue="older adult" />
        </Function>
      </Execute>
      <Execute action="action = 'a2'" ref_command="cmd1">
        <Function functionID="f2" functionName="openURL">
          <Parameter paramName="id" datavalue="1" />
          <Parameter paramName="name" datavalue="john" />
          <Parameter paramName="url" datavalue="http://example.com" />
          <Parameter paramName="textMsg" datavalue="Read the medical giudelines..." />
          <Parameter paramName="population_restriction" datavalue="older adult" />
        </Function>
      </Execute>
      <Execute action="action = 'a3'" ref_command="cmd1">
        <Function functionID="f3" functionName="dial-emergency">
          <Parameter paramName="id" datavalue="1" />
          <Parameter paramName="dept" datavalue="emergency" />
          <Parameter paramName="dialNo" datavalue="999" />
          <Parameter paramName="population_restriction" datavalue="older adult" />
        </Function>
      </Execute>
    </GetNotification>
    </Executor>
  </AA>
</AAPolicy>



 <!-- Define Events Handle by AA2 Policy -->
<AAPolicy policyID="p2" policyType="suicide" policyDescription="carer's medium help intervention" ref_aa="aa2">
<ManagedResourceList>
<ManagedResource resourceID="mr1" resourceType="suicide-attempt" ref_sensor="s1">
<TriggerEvent>
<or><Event inputEvent="suicide-attempt" op="="> yes </Event></or>
<or><Event inputEvent="suicide-attempt" op="="> no </Event></or>
<or><Event inputEvent="suicide-attempt" op="="> dont-know </Event></or>
</TriggerEvent>
</ManagedResource>
<ManagedResource resourceID="mr2" resourceType="more-than-one-suicide-attempt" ref_sensor="s2">
<TriggerEvent>
<or><Event inputEvent="more-than-one" op="="> yes </Event></or>
<or><Event inputEvent="more-than-one" op="="> no </Event></or>
<or><Event inputEvent="more-than-one" op="="> dont-know </Event></or>
</TriggerEvent>
</ManagedResource>
<ManagedResource resourceID="mr3" resourceType="total-suicide-attempt" ref_sensor="s3">
<TriggerEvent>
<or><Event inputEvent="total-attempt" op="="> 1 </Event></or>
<or><Event inputEvent="total-attempt" op="="> 2 </Event></or>
<or><Event inputEvent="total-attempt" op="="> 3 </Event></or>
</TriggerEvent>
</ManagedResource>
<ManagedResource resourceID="mr4" resourceType="suicide-note-left" ref_sensor="s4">
<TriggerEvent>
<or><Event inputEvent="suicide-note" op="="> yes </Event></or>
<or><Event inputEvent="suicide-note" op="="> no </Event></or>
<or><Event inputEvent="suicide-note" op="="> dont-know </Event></or>
</TriggerEvent>
</ManagedResource>
<ManagedResource resourceID="mr5" resourceType="seriousness" ref_sensor="s5">
<TriggerEvent>
<or><Event inputEvent="seriousness" op="="> 0.0 </Event></or>
<or><Event inputEvent="seriousness" op="="> 0.1 </Event></or>
<or><Event inputEvent="seriousness" op="="> 0.2 </Event></or>
<or><Event inputEvent="seriousness" op="="> 0.3 </Event></or>
<or><Event inputEvent="seriousness" op="="> 0.4 </Event></or>
<or><Event inputEvent="seriousness" op="="> 0.5 </Event></or>
<or><Event inputEvent="seriousness" op="="> 0.6 </Event></or>
<or><Event inputEvent="seriousness" op="="> 0.7 </Event></or>
<or><Event inputEvent="seriousness" op="="> 0.8 </Event></or>
<or><Event inputEvent="seriousness" op="="> 0.9 </Event></or>
<or><Event inputEvent="seriousness" op="="> 1.0 </Event></or>
</TriggerEvent>
</ManagedResource>
<ManagedResource resourceID="mr6" resourceType="hide-suicide-attempt" ref_sensor="s6">
<TriggerEvent>
<or><Event inputEvent="hide-suicide-attempt" op="="> 0.0 </Event></or>
```

```
<or><Event inputEvent="hide-suicide-attempt" op="="> 0.1 </Event></or>
<or><Event inputEvent="hide-suicide-attempt" op="="> 0.2 </Event></or>
<or><Event inputEvent="hide-suicide-attempt" op="="> 0.3 </Event></or>
<or><Event inputEvent="hide-suicide-attempt" op="="> 0.4 </Event></or>
<or><Event inputEvent="hide-suicide-attempt" op="="> 0.5 </Event></or>
<or><Event inputEvent="hide-suicide-attempt" op="="> 0.6 </Event></or>
<or><Event inputEvent="hide-suicide-attempt" op="="> 0.7 </Event></or>
<or><Event inputEvent="hide-suicide-attempt" op="="> 0.8 </Event></or>
<or><Event inputEvent="hide-suicide-attempt" op="="> 0.9 </Event></or>
<or><Event inputEvent="hide-suicide-attempt" op="="> 1.0 </Event></or>
</TriggerEvent>
</ManagedResource>
<ManagedResource resourceID="mr7" resourceType="wanted-to-succeed" ref_sensor="s7">
<TriggerEvent>
<or><Event inputEvent="wanted-to-succeed" op="="> 0.0 </Event></or>
<or><Event inputEvent="wanted-to-succeed" op="="> 0.1 </Event></or>
<or><Event inputEvent="wanted-to-succeed" op="="> 0.2 </Event></or>
<or><Event inputEvent="wanted-to-succeed" op="="> 0.3 </Event></or>
<or><Event inputEvent="wanted-to-succeed" op="="> 0.4 </Event></or>
<or><Event inputEvent="wanted-to-succeed" op="="> 0.5 </Event></or>
<or><Event inputEvent="wanted-to-succeed" op="="> 0.6 </Event></or>
<or><Event inputEvent="wanted-to-succeed" op="="> 0.7 </Event></or>
<or><Event inputEvent="wanted-to-succeed" op="="> 0.8 </Event></or>
<or><Event inputEvent="wanted-to-succeed" op="="> 0.9 </Event></or>
<or><Event inputEvent="wanted-to-succeed" op="="> 1.0 </Event></or>
</TriggerEvent>
</ManagedResource>
</ManagedResourceList>

<!-- Social network entities may also generate event if action sent to the network -->
<SocialNetwork networkID="network1" networkName="grace-care-network">
<NetworkResourceList>
<NetworkResource resourceID="" resourceType="" ref_sensor="">
<TriggerEvent>
<or><Event inputEvent="" op="="></Event></or>
</TriggerEvent>
</NetworkResource>
</NetworkResourceList>
</SocialNetwork>

<!-- Configure AA2 components -->
<AA adapterID="aa2" adapterType="suicide-concept-adapter">
<SensorList>
<Sensor sensorID="s1" sensorType="suicide-attempt" />
<Sensor sensorID="s2" sensorType="more-than-one" />
<Sensor sensorID="s3" sensorType="total-attempt" />
<Sensor sensorID="s4" sensorType="suicide-note" />
<Sensor sensorID="s5" sensorType="seriousness" />
<Sensor sensorID="s6" sensorType="hide-suicide-attempt" />
<Sensor sensorID="s7" sensorType="wanted-to-succeed" />
</SensorList>

<!-- Configure AA2 Monitor policy -->
<Monitor monitorID="m1">
<GetSensorData ref_sensor="s1" ref_event="suicide-attempt" />
```

```
<GetSensorData ref_sensor="s2" ref_event="more-than-one" />
<GetSensorData ref_sensor="s3" ref_event="total-attempt" />
<GetSensorData ref_sensor="s4" ref_event="suicide-note" />
<GetSensorData ref_sensor="s5" ref_event="seriousness" />
<GetSensorData ref_sensor="s6" ref_event="hide-suicide-attempt" />
<GetSensorData ref_sensor="s7" ref_event="wanted-to-succeed" />

<Condition conditionID="c1">
<Expression exprID="expr1" ref_event="suicide-attempt">
<If varName="suicide-attempt" op="==" datavalue="yes"> likely </If>
<If varName="suicide-attempt" op="==" datavalue="no"> unlikely </If>
<If varName="suicide-attempt" op="==" datavalue="dont-know"> unknown </If>
</Expression>
<Expression exprID="expr2" ref_event="more-than-one">
<If varName="more-than-one" op="==" datavalue="yes"> high </If>
<If varName="more-than-one" op="==;" datavalue="no"> medium </If>
<If varName="more-than-one" op="==;" datavalue="dont-know"> unknown </If>
</Expression>
<Expression exprID="expr3" ref_event="total-attempt">
<If varName="total-attempt" op="&gt;=" datavalue="3"> likely </If>
<If varName="total-attempt" op="&lt;" datavalue="3"> unlikely </If>
<If varName="total-attempt" op="==" datavalue="dont-know"> unknown </If>
</Expression>
<Expression exprID="expr4" ref_event="suicide-note">
<If varName="suicide-note" op="==" datavalue="yes"> unlikely </If>
<If varName="suicide-note" op="==" datavalue="no"> likely </If>
<If varName="suicide-note" op="==" datavalue="dont-know"> unknown </If>
</Expression>
<Expression exprID="expr5" ref_event="seriousness">
<If varName="seriousness" op="&gt;=" datavalue="0.5"> likely </If>
<If varName="seriousness" op="&lt;" datavalue="0.5"> harmless </If>
<If varName="seriousness" op="==" datavalue="dont-know"> unknown </If>
</Expression>
<Expression exprID="expr6" ref_event="hide-suicide-attempt">
<If varName="hide-suicide-attempt" op="&gt;=" datavalue="0.5"> hiding </If>
<If varName="hide-suicide-attempt" op="&lt;" datavalue="0.5"> no-hiding </If>
<If varName="hide-suicide-attempt" op="==" datavalue="dont-know"> unknown </If>
</Expression>
<Expression exprID="expr7" ref_event="wanted-to-succeed">
<If varName="wanted-to-succeed" op="&gt;=" datavalue="0.5"> desire-succeed </If>
<If varName="wanted-to-succeed" op="&lt;" datavalue="0.5"> no-desire </If>
<If varName="wanted-to-succeed" op="==" datavalue="dont-know"> unknown </If>
</Expression>
</Condition>
</Monitor>

<!-- Configure AA2 Local RIE policy -->
<LocalRIE rieID="aa-rie2">
<GetMonitorData ref_monitor="m1" ref_condition="c1">
<GetContextVariable contextID="context1" ref_expression="expr1"/>
<GetContextVariable contextID="context2" ref_expression="expr2"/>
<GetContextVariable contextID="context3" ref_expression="expr3"/>
<GetContextVariable contextID="context4" ref_expression="expr4"/>
<GetContextVariable contextID="context5" ref_expression="expr5"/>
<GetContextVariable contextID="context6" ref_expression="expr5"/>
```

```
<GetContextVariable contextID="context7" ref_expression="expr5"/>
</GetMonitorData>
<Condition conditionID="c2">
<Expression exprID="expr1">
<and><If varName="suicide-attempt" op="==" datavalue="likely" ref_context="context1"> true </If></and>
<and><If varName="more-than-one" op="==" datavalue="high" ref_context="context2"> true </If></and>
<and><If varName="total-attempt" op="==" datavalue="likely" ref_context="context3"> true </If></and>
<and><If varName="suicide-note" op="==" datavalue="unlikely" ref_context="context4"> true </If></and>
<and><If varName="seriousness" op="==" datavalue="likely" ref_context="context5"> true </If></and>
<and><If varName="hide-suicide-attempt" op="==" datavalue="hiding" ref_context="context6"> true </If></and>
<and><If varName="wanted-to-succeed" op="==" datavalue="desire-succeed" ref_context="context7"> true </If></and>
</Expression>
<Expression exprID="expr2">
<and><If varName="suicide-attempt" op="==" datavalue="unlikely" ref_context="context1"> true </If></and>
<and><If varName="more-than-one" op="==" datavalue="low" ref_context="context2"> true </If></and>
<and><If varName="total-attempt" op="==" datavalue="unlikely" ref_context="context3"> true </If></and>
<and><If varName="suicide-note" op="==" datavalue="likely" ref_context="context4"> true </If></and>
<and><If varName="seriousness" op="==" datavalue="unlikely" ref_context="context5"> true </If></and>
<and><If varName="hide-suicide-attempt" op="==" datavalue="no-hiding" ref_context="context6"> true </If></and>
<and><If varName="wanted-to-succeed" op="==" datavalue="no-desire" ref_context="context7"> true </If></and>
</Expression>
</Condition>

<TriggerAction ref_condition="c2">
<While condition="expr1 = 'true'">
<ActionTemplate actionID="a1" actionType="urgent-support" category="none" target_resource="999">
galassify-action-urgent-support-template </ActionTemplate>
</While>
<While condition="expr2 = 'true'">
<ActionTemplate actionID="a2" actionType="send-sms" category="none" target_resource="patient-node">
galassify-action-send-sms-template </ActionTemplate>
</While>
</TriggerAction>
</LocalRIE>

<!-- Configure AA2 Executor policy -->
<Executor executorID="exe1">
<GetNotification message="yes" ref_sender="globalRIE" >
<Execute action="action = 'a1'" ref_command="cmd2">
<Function functionID="f1" functionName="urgentSupport">
<Parameter paramName="id" datavalue="1" />
<Parameter paramName="dept" datavalue="emergency" />
<Parameter paramName="recepient" datavalue="abc@email.com" />
<Parameter paramName="telNO" datavalue="999" />
<Parameter paramName="textMsg" datavalue="Need urgent support..." />
<Parameter paramName="population_restriction" datavalue="older adult" />
</Function>
</Execute>
<Execute action="action = 'a2'" ref_command="cmd2">
<Function functionID="f2" functionName="sendSMS">
<Parameter paramName="id" datavalue="1" />
<Parameter paramName="name" datavalue="john" />
<Parameter paramName="mob" datavalue="077993344" />
<Parameter paramName="textMsg" datavalue="Read the medical giudelines..." />
<Parameter paramName="population_restriction" datavalue="older adult" />
```

```
</Function>
</Execute>
</GetNotification>
</Executor>
</AA>
</AAPolicy>

<!-- Configure Global Policy -->
<GlobalPolicy>
<GlobalRIE rieID="globalRIE">
<ActionList>
<!-- Assume that send-email action triggered by AA1 -->
<GetAction ref_aa="aa1">
<or><Action inputActionID="aa1-action"> a1 </Action></or>
<or><Action inputActionID="aa1-action"> a2 </Action></or>
<or><Action inputActionID="aa1-action"> a3 </Action></or>
</GetAction>
<!-- Assume that some actions may triggered by AA2 -->
<GetAction ref_aa="aa2">
<or><Action inputActionID="aa2-action"> a1 </Action></or>
<or><Action inputActionID="aa2-action"> a2 </Action></or>
</GetAction>
<GetAction ref_aa="aa3">
<or><Action inputActionID="aa2-action"> a1 </Action></or>
<or><Action inputActionID="aa2-action"> a2 </Action></or>
</GetAction>
</ActionList>

<Condition conditionID="c3">
<Expression exprID="expr1">
<and><If varName="action" op="==" datavalue="a1" ref_input_action="aa1_action"> true </If></and>
<and><If varName="action" op="==" datavalue="a1" ref_input_action="aa2_action"> true </If></and>
<and><If varName="action" op="==" datavalue="a2" ref_input_action="aa3_action"> true </If></and>
</Expression>
<Expression exprID="expr2">
<and><If varName="action" op="==" datavalue="a2" ref_input_action="aa1_action"> true </If></and>
<and><If varName="action" op="==" datavalue="a2" ref_input_action="aa2_action"> true </If></and>
</Expression>
<Expression exprID="expr3">
<and><If varName="action" op="==" datavalue="a3" ref_input_action="aa1_action"> true </If></and>
<and><If varName="action" op="==" datavalue="a2" ref_input_action="aa2_action"> true </If></and>
</Expression>
<Expression exprID="expr4">
<and><If varName="action" op="==" datavalue="a3" ref_input_action="aa1_action"> true </If></and>
<and><If varName="action" op="==" datavalue="a2" ref_input_action="aa2_action"> true </If></and>
</Expression>
</Condition>
<Decision ref_condition="c3" ref_expression="expr1">
<Command commandID="cmd1" message="yes" target_adapter="aa1"> a1 </Command>
<Command commandID="cmd2" message="yes" target_adapter="aa2"> a1 </Command>
<Command commandID="cmd3" message="yes" target_adapter="aa3"> a1 </Command>
</Decision>
<Decision ref_condition="c3" ref_expression="expr2">
<Command commandID="cmd1" message="yes" target_adapter="aa1"> a2 </Command>
<Command commandID="cmd2" message="yes" target_adapter="aa2"> a2 </Command>
```

```
</Decision>
<Decision ref_condition="c3" ref_expression="expr3">
<Command commandID="cmd1" message="yes" target_adapter="aa1"> a3 </Command>
<Command commandID="cmd2" message="yes" target_adapter="aa2"> a2 </Command>
</Decision>
</GlobalRIE>
</GlobalPolicy>
</PolicySet>
```