Policy Based Generic Autonomic Adapter For A Context-Aware Social-Collaborative System

Nazmul Hussain
Engineering and Applied Science
Aston University
Birmingham, B4 7ET
Email: hussan12@aston.ac.uk

Hai H Wang
Engineering and Applied Science
Aston University
Birmingham, B4 7ET
Email: h.wang10@aston.ac.uk

Christopher Buckingham
Engineering and Applied Science
Aston University
Birmingham, B4 7ET
Email: c.d.buckingham@aston.ac.uk

***Abstract*—Autonomic computing was intended to tackle the growing complexity of Information Technology infrastructure by making it self-managing and self-adaptive. The core idea is to endow the system with enough intelligence to monitor continuously all aspects of the changing environments and resources, and to control management decisions according to high-level policies. For several years, great efforts have been devoted to the study of system performance, security, and fault management issues, but without much attention paid to self-adaptive social-collaborative system development. This may be because it is difficult to create such autonomic systems, which must sense and adapt to ongoing social context changes and support cyber-physical collaborations with minimal human involvement. These collaborations will have interactions between human and non-human entities that need to be self-managing with adaptive goals.**

This paper tackles the problem by introducing a new Generic Autonomic Social-Collaborative Framework (GASCF). It focuses on a high-level social-context based self-adaptive system, and its use of intelligent agents called autonomic adapters(AAs) that are driven by predefined policies. The paper describes the architecture of autonomic adapters and the general representation of policies. It explores the effectiveness of the approach by applying it to a large-scale collaborative healthcare service called GRaCE (https://www.egrist.org/) that supports mental-health within the United Kingdom National Health Service and other organisations.

***Index Terms*—Autonomic Computing, Intelligent Agent, Context-aware system, ECA Policy, Social Network, Healthcare, GRaCE mental health.**

I. INTRODUCTION

Autonomic Computing (AC) was introduced by IBM in 2001 [1] with the vision to tackle the growing complexity of Information Technology (IT) infrastructure by making systems and applications self-managing. Autonomic systems have been extended by the capability of adapting themselves based on context-awareness features [2]. Low-level decision making is often encapsulated by the well-known Event-Condition-Action (ECA) rule-based policy model [3]. The conditions of rules are well suited to detecting changing contexts so that the actions can provide relevant services and information for users. Systems are considered socially context-aware if they are aware of the users' concerns, goals, and social norms, and are able to act on behalf of them [4].

978-1-5386-4396-9/18/$31.00

Self-adaptive software aims to adjust various artifacts or attributes in response to changes in run-time and in the context of the operating environment [5]. Several approaches have been adopted to tackle the adaptation in different levels and domains to maintain system functionality. However, there are some important targets and goals that autonomic computing still needs to address. For example, in the autonomic network, manually configured devices do not have the capability to share their current status with their neighbouring devices [6]. A study reported that specific autonomic elements must be designed with a greater awareness of the fact that they will be installed in autonomic systems and intercommunicating and interacting cooperatively with other autonomic elements [7]. Additionally, previous adaptation work is done in an ad-hoc manner, which is based on predicting future circumstances and embedding the adaptation decisions in the program code [8]. Instead, policy should be specified to control the adaptation behaviour in a more flexible way, outside the code itself.

The AC approach has been successfully applied in many disciplines such as autonomic space exploration missions [9], autonomic software applications [10], and scientific collaboration [11]. Social networking is a more recent domain where it would seem to be particularly appropriate [12]. The socially context-aware system aims to support human communication and collaboration based on sensed activity and interaction with the physical world [13]. People do not just connect to each other, they also connect through shared objects or artifacts [14]. Existing social software solutions focus on maintaining human to human interaction only, whereas non-human factors are also needed for realising the full potential of context-aware autonomic service invocation.

A common issue is that the social network environment constantly grows, and the context changes as new information and content are created and stored, which requires monitoring and re-analysis for each change [12]. A similar problem occurred in telephony in 1920 due to the rapid expansion of phone usage, which raised a real concern that there would not be enough trained operators to control the switchboard manually [15]. Fortunately, automatic branch exchanges were put in place to eliminate the need for human intervention.

Likewise, an AC approach can be employed to improve collaborative information processing in the social network so

that system-wide self-management is achieved [16]. However, robust autonomic management requires the managing agents to be intelligent enough to monitor and control their decisions [4]. In order to perform self-adaptation, autonomous agents must rely on knowledge to decide why, when, how and where to perform the adaptation operations [17].

To address these issues, we propose a Generic Autonomic Social-Collaborative Framework (GASCF) for developing a socially context-aware collaborative system that will be self-managed and adaptive according to the application domain context. It is founded on individual and generic Autonomic Adapters (AAs) with their behaviour governed by ECA rule-based policy. We demonstrate the potential of this approach by applying it to the GRaCE mental-health risk and safety management decision support system [18], [19], [20] that is being used within the English National Health Service and other organisations. We then discuss this work in terms of its overall contribution and what is necessary to evaluate and compare such systems in the future.

The remainder of the paper is organized as follows: Section II describes some background to autonomic computing and social-collaborative systems in general. Section III introduces the GASCF with a generic AA architecture and general-purpose policy specification language. Section IV creates a generic policy-specification language for the GASCF and Section V explores how the GASCF can be applied to the GRaCE mental-health collaborative system. Section VI discusses the originality of the proposed solution and Section VII concludes with further research directions.

II. BACKGROUND AND MOTIVATION

This section reviews state-of-the-art autonomic computing systems. It considers how social networking system can be applied as communication tools for self-adaptive collaborative systems and also provides the motivation for designing our frameworks that implement them.

A. Motivation

A prime motivator for this paper is the suitability of the proposed system within healthcare, which is founded on social-based collaborative work, where multiple people work together to provide care for patients in hospital or at home [21]. Remote monitoring is considered an essential part of future e-Health systems, to enable delivery of care support outside clinical sites at a reduced cost while improving quality of patient-care [22]. Healthcare is delivered in the context of a resource-intensive physical environment, which is often structured to facilitate collaborative work such as care support and is organised across different workers through the use of artifacts [21]. There is a continuous need for self-management across all of these care environments with little or no user input. More recently, ensuring the remote healthcare system stays in tune with care needs has generated much interest in autonomic computing [23], [24]. Autonomous adaptation of behaviour according to the patient's body, status of related physical devices, and the social context of human interactions would make major improvements to the holistic and evolving care needs of modern health services.

B. Autonomic Computing Overview

Autonomic computing (AC) is an emerging paradigm inspired by the human Autonomic Nervous System that regulates the body using parameters such as heart rate or temperature [25]. From a technology perspective, its aim is to control the functioning of computer applications and systems without human intervention by using an intelligent control called the MAPE-K loop (monitoring, analysing, planning and executing functions in conjunction with a shared Knowledge-base). This was proposed by IBM in 2003 in its architectural blueprint for autonomic computing [26]. The resources managed and controlled in the MAPE-K loop constitute the smallest functional unit that exists in the run-time IT environment such as CPUs, databases, web servers, legacy applications, etc.

In theory, AC encompasses four self-management capabilities [1], [27]: *self-configuring* is the ability to adapt and reconfigure automatically in response to the changing environments according to high-level goals; *self-healing* is the ability to detect errors at runtime and initiate policy-based corrective action without disruption of system services; *self-optimizing* is the ability to monitor and tune resources automatically; and *self-protecting* systems should have the ability to anticipate, detect, identify, and protect themselves from attacks from anywhere.

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 [1]. This requires an accurate model of knowledge specification, acquisition and processing [17]. For example, rule engines and correlation engines (with their associated languages) are useful technologies for analysing monitored data and log files to identify trends or situations that warrant deeper examination [7].

Policy-based management using Event-Condition-Action (ECA) rules [3] 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* [3]. 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.

C. Autonomic Social-Collaboration Environment

Social networking software creates a new opportunity to initiate collaborative work and help people maintain contact, be aware of the current situation, and communicate [14]. Collaboration is the process of working together in a sociable environment to systematically solve a problem that could not be solved by an individual alone. Nowadays, many social networking tools (e.g. wikis, blogs, Facebook, etc.), facilitate collaboration by providing a social context [28]. Traditionally, email was used for collaboration because of its flexibility and ease of use in notifying people. However, despite people using it more recently in a quasi-synchronous manner [29], it

is not truly real-time. Instead, social networks employ event notifications.

A typical notification policy is that when an event happens at a node, the event is reported to all nodes that are directly connected to the source node. However, propagating events without overwhelming people with relevant information or alerts is an open research question [14]. Policy-based autonomic computing will address it by supporting human and non-human collaboration through a notification mechanism that uses continuous acquisition and analysis of large-scale and dynamic event data from the physical objects and social network. An important element of the mechanism will be who needs to know what and when, which comes from a clear understanding of the social context.

III. DESIGN OF THE GENERIC AUTONOMIC SOCIAL-COLLABORATIVE FRAMEWORK (GASCF)

In this section, a description of the proposed GASCF and its generic Autonomic Adapter (AA) architecture is presented.

A. High-Level Architecture of the GASCF

The GASCF is a web-based framework, which facilitates the autonomous processing of event data from various sources, such as the situation of physical devices, social artifacts or human social interactions. This framework can be used to implement a context-aware social-collaborative application for a particular domain that will enable self-managed and adaptive functionality. On this platform, multiple AAs will work in cooperation to monitor the system's behaviour, keep track of changes and make desired alterations to the target hardware and software resources, or send notifications to network nodes according to predefined policies.

The set of managed resources (e.g., controlled system component [26]) and AAs together form the logical structure of an intelligent collaborative system that communicates through a notification mechanism to coordinate decisions. Basically, the AAs in the GASCF have two responsibilities: firstly, they monitor, acquire, and utilise the resources' status of the execution environment; and secondly, in case of significant contextual changes in the execution environment, they adapt the affected applications or resources. Figure 1 describes the high-level architecture of GASCF, which incorporates three main separate layers.

i) Application Layer is the cyber-physical environment for the connected IT infrastructure where real-world collaborative work takes place. It may consist of physical sources such as hardware devices and sensors, and human beings that interact with the social networking system.

ii) Autonomic Computing Layer is a collection of interactive AAs (e.g., AA_1, AA_2 ... AA_n), which manages one or more managed resource (MR) (e.g., MR_1, MR_2 ... MR_n) each implementing a relevant intelligent loop (e.g., Monitor (M), Decision Maker (DM) and Executor (E)) that controls the functioning of computer applications and systems in conjunction with a shared Knowledge (K). The AAs runs autonomously and cooperate with each other as a whole to

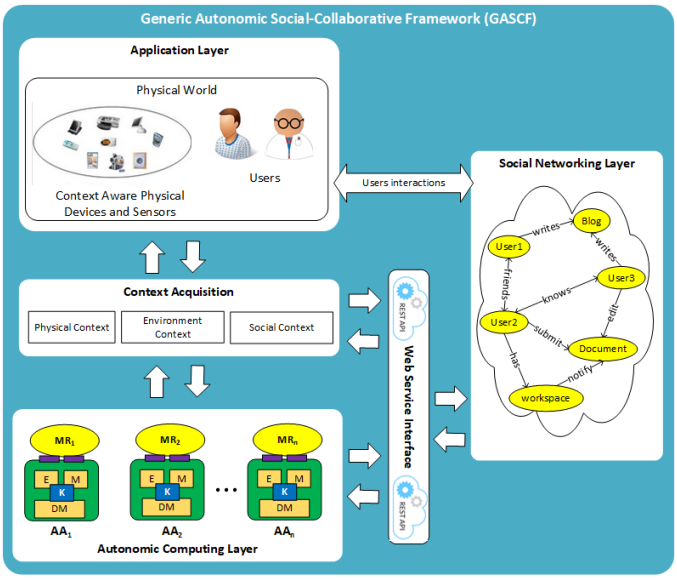

Fig. 1: High-level Architecture of GASCF

produce self-managing and adaptive behaviour. These managed resources could be physical objects, software elements or network resources. They are attached to AAs and communicate through a set of web service interfaces. AAs are responsible for monitoring, gathering and aggregating raw context data about an event from various distributed sources. They analyse the data and determine the actions needed for the system. Decisions will be taken based on the overriding policies, which are configured according to predictable context changes by domain experts. Context acquisition is related to users' physical, environmental, and social context. For instance, user-related contextual information may include physiological conditions (e.g., heart rate, blood pressure, and emotional state), social situations relate to the connections between people and their interactions, and environmental factors involve location, time, light, and room temperature amongst others.

iii) Social Networking Layer is the environment that represents all the social factors and relevant resources (e.g., users, blogs, documents) and relationships (e.g., friends, knows, notifies), which can provide a social-collaborative context in the real-world application domain. Users can directly interact with the social network, and integrated AAs will monitor events and take actions accordingly. For example, update a document and send notifications to the people stated in the policy scripts for when that document is updated.

The ***Web Service Interface*** is simply a RESTful API implemented as application drivers that provide a standard communication protocol within GASCF over the Web Servers and implement the query processing capability to send and retrieve data.

B. Generic AA Architecture

AAs are the fundamental intelligent agents of the GASCF that perceive input events from their managed resources and use these percepts to determine actions to execute through

their effectors. An action may raise further events during its execution. They are intelligent because they regulate their own internal states and facilitate collaborative interaction with other AAs externally through event notification and knowledge utilisation. The example resources managed by an AA can be a hardware entity, a software entity, any type of network resources, the behaviour of a particular metric or service, or a network as a whole. Figure 2 illustrates the generic AA architecture.

An individual AA builds upon four main functional elements: a sensor interface, an intelligent manager loop, a rule inference engine, and an intelligent policy evaluator:

i) Sensor Interface: An AA can control `events` from one or more *managed resources* through its several *sensors*. Each *sensor* detects and signals the occurrence of `events` from numerous *managed resources* in the IT environment and sends `sensor data` to the *monitor* component. Log adapter and profiling is likely to be the simplest technique for capturing information from *managed resources*.

ii) Intelligent Manager: In the intelligent manager loop, the *monitor* component is responsible for collecting and correlating `sensor data` data received from multiple *sensors*. The `result` is sent to the *decision maker*, which is responsible for extracting and aggregating monitoring data. It determines whether some changes need to be made, learns about the application environment and constructs the actions needed to achieve a certain goal. The `actions` usually indicate some sort of alteration to one or more *managed resource* according to some guiding `policy or strategy` as well as sending an action-list to the *execute* component. Two sub-components `(analysis and plan)` cooperatively perform decision making function. The *execute* component receives a logical description of the sequence of `actions` to be executed. It provides the mechanisms to control the execution of the `plan` over the *managed resources* or IT environment based on a set of design patterns that allow the software system to change some artifacts during run-time.

iii) Rule Inference Engine (RIE): The RIE is a component of the system that applies logical rules to the knowledge base to deduce new information. The AAs functionality is driven by the `policy`, which is triggered and interpreted as `shared knowledge` in the RIE. Knowledge in general can be almost any sort of structured data or information that could be used to carry out processes when changes need to be applied to a system. For example, knowledge about past states, event facts, action vocabularies, policy knowledge, etc. It is not specific to any one AA and so is contained in a shared central resource "RIE"outside the AAs but with which all AAs can communicate.

iv) The Intelligent Policy Evaluator (IPE): The IPE plays the role of a central mediator between AA components and the policy evaluation functionality. It is the mechanism that implements the `production rules` associated with the specific AA that determine what `actions` should be executed when relevant `events` are observed. The complete `set of rules` constitute the system `policy` that lies outside all the AAs but is shared between them. In the IPE, *Condition Evaluator* selects the relevant `rules` for its own AA by finding those that having matching `conditions` to the factors specified by the *decision maker*. The *Action Scheduler* organises the triggered policies to produce the schedule of actions for execution.

C. Process flow of An Autonomic Adapter

Each AA's intelligent manager component has a separate function that takes inputs and produces outputs as shown in Figure 2. Sensors are connected to one or more managed resources and configured to collect specific data from each one. These data are then streamed in real time to all AAs with which a sensor is connected, so the sensors do not themselves know or care about whether output data is of interest to a connected AA or not.

The monitor component of an AA is configured to respond to specific elements of the sensor data and is where the relevance of sensor data is determined. Data may come from multiple sensors and the monitor's job is to specify which factors are of interest and whether this is only for particular value ranges. When factors and ranges meet the parameters of the monitor, outputs are sent to the analysis component. This connects to the system policies and finds all the policy rules that have conditions matching the monitor outputs. The actions for those policies are collated and organised by the plan component and passed to the execution component, which carries out the plan by updating associated elements of the system environment. In general, managed resources will be updated by the AA that changes the environment, triggering different sensor outputs that cause the set of AAs to respond in an appropriate and updated way. This is the heart of autonomic adaptation, where the central policies determine how adaptation takes place.

1) Event Notifications Workflow: Communication between AAs uses event notifications. In this context, an event is generated whenever data fall within the interest of AA components, starting with the sensors. When an AA's sensors receive data falling within their remit, an event is triggered and the data is sent to all the associated AAs. The AA monitor receives the data and if the set contains variables and values matching the monitor's parameters, an event is triggered and the matching data set is sent to the analysis component. The analysis component triggers an event when it finds all matching policies and produces an action plan accordingly. This action plan is sent to the execution module and triggers the execution event, by which the plan updates the environment and managed resources as specified. This final event may or may not trigger new events in sensors that motivate the next iterations. Note that there is no explicit linkage of AAs: they interact solely by sharing sensors and reacting to the same environment.

IV. POLICY SPECIFICATION LANGUAGE FOR AUTONOMIC ADAPTERS

In this section, we describe the general-purpose Policy Specification Language (PSL) for AAs based on the well-accepted

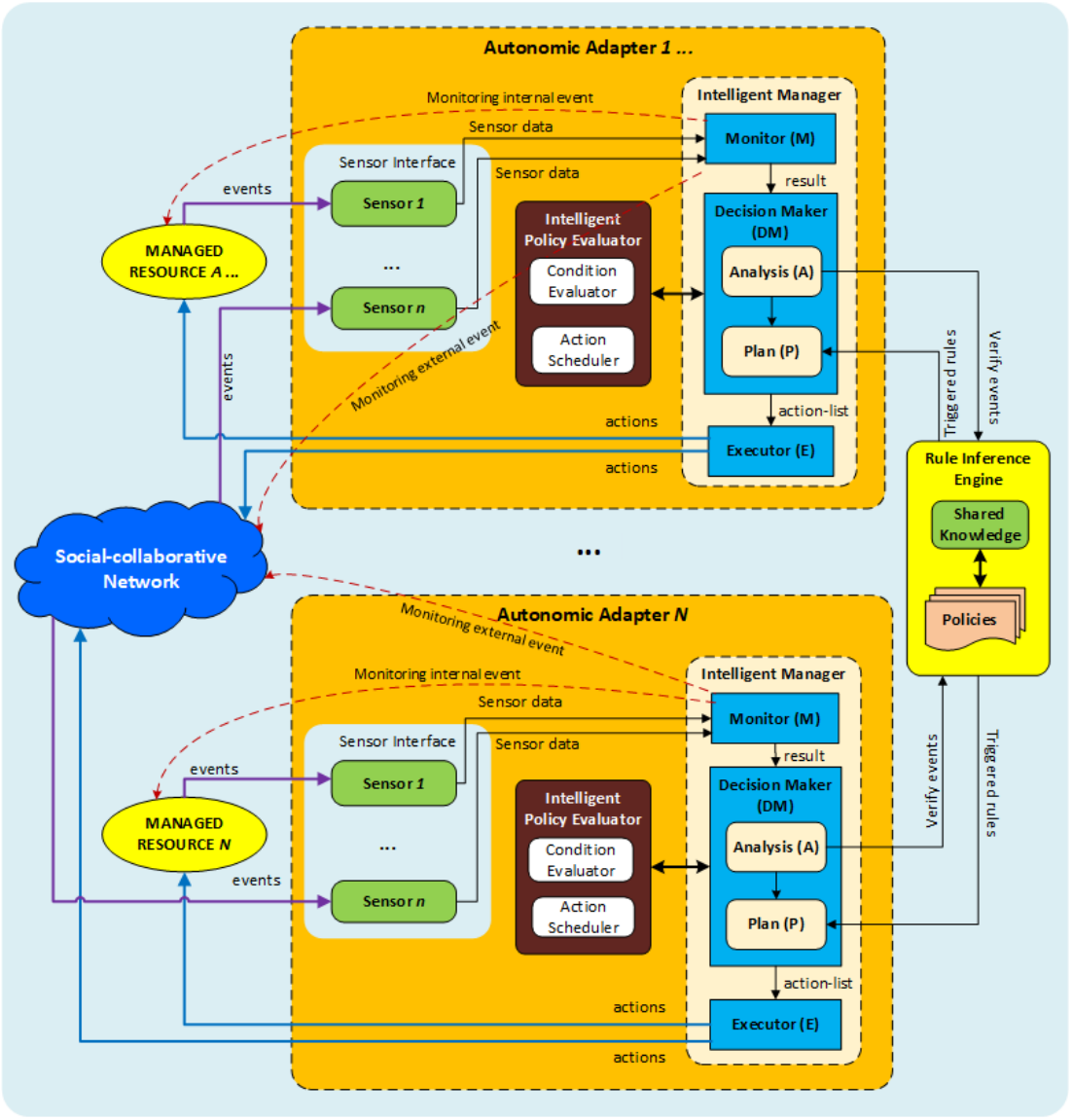

Fig. 2: Generic Autonomic Adapter Architecture

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.

A. PSL Syntax

The PSL syntax uses the Backus-Naur Form (BNF) [1] notation as follows:

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

<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'.
- 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.

V. CASE STUDY

In this section, we discuss how AAs can be applied to implement a self-adaptive and socially context-aware healthcare system. It will be based on GRaCE domain, which is a web-based software system originally called GRiST [2], used by mental health services in England.

A. GRaCE-AGE Scenario

The GASCF supports the design and development of a configurable framework focused on virtual communities for older adults in the community. It uses the older adult version of GRaCE called GRaCE-AGE and the myGRaCE, self-assessment [30], which creates a "canopy of care" functionality directly linking doctors, clinicians, care providers, and family members to the older adults to ensure they are safe, secure, and thriving. The social care network provides social tools to share health-related information, care policy and practice issues and to educate and interact with patients to provide medical support.

The resources across the GRaCE-AGE domain include a variety of physical devices and sensors, software entities, and human participants, all of which interact, communicate and send messages in different ways. AAs are attached to the individual resources to monitor their status, analyse the contextual changes, and plan and execute appropriate actions based on the defined policy. Threshold triggers are set in the AA policy for certain bio-data types such as heart-rate, blood pressure, and temperature to generate alert notifications as required. For example, if the heart rate exceeds or drops below a threshold set by the doctor, then a doctor, caregiver and a family member can be alerted to contact the patient urgently as illustrated by Figure 3.

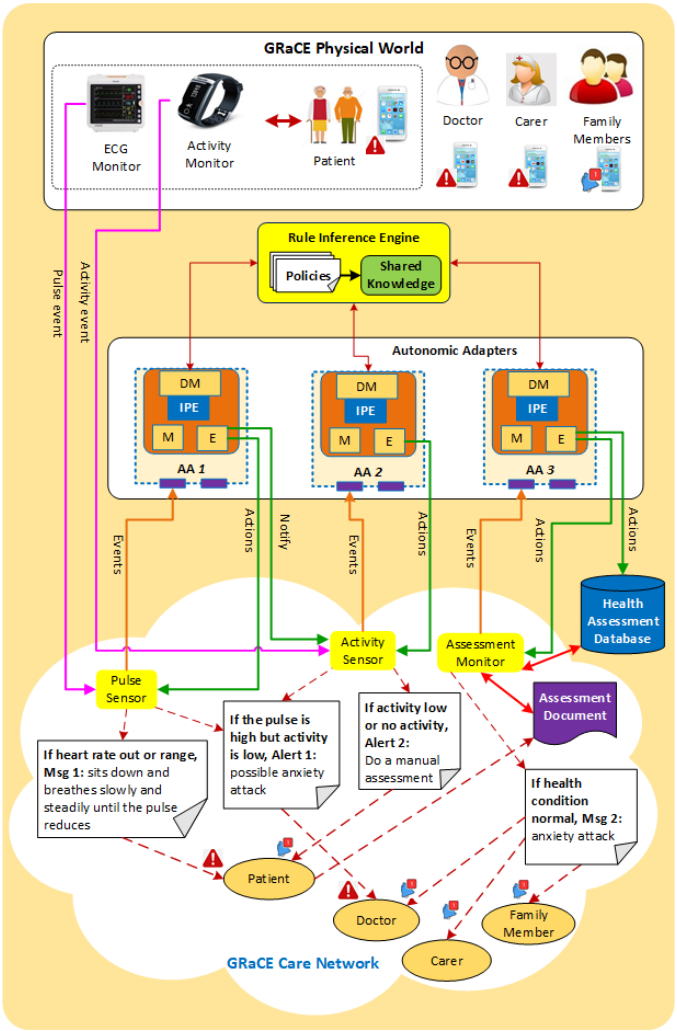

Fig. 3: GRaCE-AGE Scenario

In this scenario, one or more wearable devices are attached to a patient's body to record heart rate and physical activity. The data is sent to the GRaCE-AGE system via a set of RESTful APIs. The managed resources across the GraCE care network include various software entities that may or may not

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

be associated with physical devices: *"Pulse Sensor"*, *"Activity Sensor"*, and *"Assessment Monitor"*. AA_1 and AA_2 link to physical devices: ECG monitor and Activity monitor through managed resources *"Pulse Sensor"* and *"Activity Sensors"* respectively, but AA_3 links to a software a sensor *"Assessment Monitor"* (more like a daemon) that notifies it whenever a new assessment has been submitted to the GRaCE-AGE database.

Suppose the pulse sensor has gone over a threshold limit so that its value is within the range of interest for the associated AA, AA_1. AA_1 will access the policy knowledge and one of the fired rules will send a message directly to the person that suggests he or she sits down and breathes slowly and steadily until the pulse reduces. The same rule, or a related one, may also be fired that is the input to another rule that combines with the activity monitor. This rule will be triggered if the pulse output and the activity output are both within a given range suggesting the person is having an anxiety attack, because the pulse is high but activity is low. The output of the rule combining the two AAs will be additional advice about managing the acute anxiety episode because now more is known about it. Finally, the AA_3 may have been triggered for a new GRaCE assessment for the older adult that falls within the timespan of the anxiety attack. Data from this assessment can be combined with the outputs of the triggered rules from AA_1 and AA_2 that determines whether the care network should be alerted for the current anxiety attack.

B. Adaptation Policies

The fundamental decision making policy for individual AA's based on the above scenario is shown below in ECA syntax:
AA_1 Policy: ON (pulseChange(ManagedResource pulseSensor))
IF(pulseSensor.heartRate>140bpm)
Do {(notify(Person patientID, "sits down and breathes slowly and steadily until the pulse reduces"))
(notify(ManagedResource activitySensor))}

AA_2 Policy: ON (getNotified(ManagedResource activitySensor))
IF ((activitySensor.physicalActivity=="low") AND (pulseSensor.heartRate=="high"))
Do {(alert(Person patientID, "do a manual assessment"))
(alert(Person doctorID, "possible anxiety attack"))}

AA_3 Policy: ON (newAssessment(ManagedResource assessmentMonitor))
IF ((assessmentDocument.submission=="true") AND (assessmentEvaluation=="normal"))
Do {(notify(Person doctorID, "anxiety attack"))
(notify(Person carerID, "anxiety attack"))
(notify(Person familyMemberID, "anxiety attack"))}

The AA_1 policy is triggered when the AA's sensor component detects the heart rate and the condition ascertains the threshold parameter is greater than 140 bpm. It then calls the methods *(notify(Person patientID, "sits down and breathes slowly and steadily until the pulse reduces"))* and *notify(ManagedResource activitySensor)*. It means a message is directly sent to the patient and it also causes AA_2 to be triggered upon receiving an event notification from the managed resource *ActivitySensor*. The AA_2 policy states that if the physical activity is low and the heart rate is high, then call the methods *(alert(Person patientID, "do a manual assessment"))* and *(alert(Person doctorID, "possible anxiety attack"))*. The AA_3 policy is triggered upon detecting a new assessment document submission event from the managed resource *AssessmentMonitor*. The AA_3 policy states that if the document submission is true and the assessment values are consistent with an anxiety attack, then call the methods *(notify(Person doctorID, "anxiety attack"))*; *(notify(Person familyMemberID, "anxiety attack"))*; and *(notify(Person carerID, "anxiety attack"))*.

C. Policy Scripts

In practice, policy scripts are formatted in XML with an enforced general standard syntax for the PSL. The Example policy scripts for AA_1 is shown in Figure 4.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<PolicyFile xmlns = "http://test.com"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation = "http://test.com">
 <PolicySet ManagedResource="pulse_sensor" AdapterID="AA1">
   <Policy policyID="r1" policyName="heart rate policy"
   policyType="pulse sensor policy">
     <Event eventID="e1" source="ECG Monitor" timestamp=
     "date/time" description="heart rate out of range" />
     <Condition var="heart_rate" op=">" value="140" />
     <Actions>
       <Action actionID="1" actionName="notify activity
       sensor">
        <Method>
           <MethodSignature>notify</MethodSignature>
           <Parameter>activitySensor</Parameter>
        </Method>
        <Method>
           <MethodSignature>alert</MethodSignature>
           <Parameter>patientID</Parameter>
           <Parameter>sits down and breathes slowly and
           steadily until the pulse reduces</Parameter>
        </Method>
       </Action>
     </Actions>
   </Policy>
 </PolicySet>
</PolicyFile>
```

Fig. 4: Sample AA_1 Policy Script

VI. RELATED WORK

There have been a number of research efforts in both academia and industry to tackle adaptation in different levels and domains to maintain autonomic systems functionality. In the first place, there have been several projects that describe autonomic system infrastructure and define the architectural aspects and challenges of autonomous system in general. Initially, much of the discussion on autonomic computing often focused on servers, networks, databases, and storage management system development. More recently, Gauvrit et.al. [31] designed a framework to bring self-adaptability to service-based distributed applications. Kang et.al. [16] employed

autonomic computing to improve collaborative information processing and data aggregation in Wireless Sensors Networks to reduce expensive communication costs for large amounts of data. Silva et.al. [12] suggested that social software users would also benefit greatly from the introduction of autonomic features. They applied autonomic computing techniques to improve social network analysis that consists primarily of identifying social entities and their relationships. They argued that autonomic computing could help with the analysis and in decision-making, transforming the network into an intelligent social network.

Yan Xiao [21] reviewed field studies in healthcare and other domains for the role of physical artifacts in collaborative work and showed how artifacts are used and exploited to facilitate collaborations. His work suggested that the design and deployment of new technology should support the functions of physical artifacts in collaborative work by embedding ICT into the existing infrastructure of physical artifacts. Almomen et.al. [23] examine 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. Harroud et.al. [32] used policy-based reasoning for Agents with classic context categories (e.g., time, space, location, identity, and a service-to-user mapping).

VII. DISCUSSION

In the last few years, there has been a thriving interest in both academia and industry to develop autonomic systems in a bid to overcome growing system complexities. Kephart and Jeffrey [7] pointed out that autonomic elements will draw upon a number of common technologies, including monitoring, event correlation, rule execution, modelling, optimization, forecasting, planning, feedback control, and machine learning. Although some work has been done in the area of context monitoring, most of the approaches have been tightly coupled with the applications for which they have been developed. Moreover, major context-aware social networking platforms and applications proposed are not self-adaptive.

This paper is a modest contribution to the ongoing discussions about developing a socially context-aware autonomic system, exposing especially the self-management and self-adaptive characteristics. In this work, particular attention has been paid to designing an architecture for a generic autonomic adapter, which is easy to implement and universal enough to be applied by the GASCF to most cases. We have argued that the autonomic properties are achieved through AAs that monitor the resource states and the status of the social network nodes, automatically transform context information into operations and suggest actions based on policies defined by the domain expert. As further motivation for this research, the healthcare domain has been considered as a prime example of social-based collaborative work that needs context-aware autonomic management. We showed that AAs can offer continuous monitoring of a patient's health condition, the collection of patient data, and the ability to post information on the GRaCE care network, including triggering alarms when critical conditions are detected so that doctors or carers can instantly react to them.

Our AA model has several significant advantages over existing approaches to context-aware autonomic system development. First, it is generic, because it can be re-configured for use with any legacy or future IT system whose behaviour can be triggered on the fly without disrupting the system operation. Second, the decisions taken by the autonomic adapters are based on a rich and flexible set of high-level ECA rule-based policies that are unavailable in existing context-aware autonomic solutions. Policy scripts are formatted in XML, which enforces a general syntax of the PSL proposed in this paper. Finally, it has an intelligent policy evaluator, which runs autonomously and cooperates with each AA's component, constantly communicating with policy and knowledge sources.

As in any other autonomic system, it is also important to evaluate and compare the GASCF and its AAs against existing solutions. However, there has not been any comprehensive work addressing evaluation criteria or metrics for self-adaptive systems [5]. This is partly because testing dynamically adaptive systems is extremely challenging; the structure and behaviour of the system may change during its execution [33]. Normally, a combination of quantitative and qualitative methods is used to evaluate every characteristic or property of the system. McCann and Huebscher [34] 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 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 in relation to each adaptation process [5].

VIII. CONCLUSIONS AND FUTURE WORK

Autonomic computing systems consist of multiple autonomic elements that can be scaled to any degree. Each of them expresses different objectives and optimization criteria. The goal is to reduce system complexity by applying policy-based self-adaptive mechanisms.

We presented the GASCF as a promising approach for developing a socially context-aware collaborative application. We introduced an intelligent autonomic adapter (AA), which is an agent monitoring the status of physical resources and collaborative artifacts and which suggests actions based on policies defined by the domain experts. A policy specification language has been presented with the generic capabilities to encapsulate policies in diverse domains and control AAs accordingly. We show that the object-oriented nature of the policy specification enables highly expressive policy logic using a simple and consistent syntax. We argue that AAs can transform traditional social networks into an intelligent collaborative network capable of autonomous analysis and decision making towards one or more set goals.

The GRaCE-AGE case study demonstrates the potential for our generic approach to socially context-aware system development within a specific domain. The GASCF could enhance this real-world mental-health software system so that it can more effectively empower people to look after their own health at home, in conjunction with their care network, with a concomitant reduction in load on overburdened health services.

One of the paramount challenges that autonomic systems research is currently confronting relates to describing the appropriate knowledge model. To streamline the process, some research recommends storing knowledge in an ontology, to benefit from the straightforward formalism of first-order predicate logic. This is a promising research direction, given the continuous expansion of the Semantic Web and its close connection with ontology-based knowledge representations. More work is needed on how to exploit a formal knowledge model within the intelligent loops, and on the addition of more configurable AA application program interfaces in order to support domain-specific needs. Our paper has provided powerful motivation for pursuing this programme of research to increase the power and reach of autonomic systems.

Acknowledgements

This work was supported and partially funded by the KIC EIT Health Grant for GRaCE-AGE.

REFERENCES

[1] P. Horn, "Autonomic computing: Ibm's perspective on the state of information technology," 2001.

[2] E. Mezghani, R. B. Halima, I. B. Rodriguez, and K. Drira, "A model driven approach for automated design of context-aware autonomic architectures," 2012.

[3] J. Bailey, A. Poulovassilis, and P. T. Wood, "An event-condition-action language for xml," in *Proceedings of the 11th international conference on World Wide Web*. ACM, 2002, pp. 486–495.

[4] C. Kennedy, "Decentralised metacognition in context-aware autonomic systems: Some key challenges." in *Metacognition for Robust Social Systems*, 2010.

[5] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," *ACM transactions on autonomous and adaptive systems (TAAS)*, vol. 4, no. 2, p. 14, 2009.

[6] A. Louca, A. Mauthe, and D. Hutchinson, "Autonomic network management for next generation networks," *PG Net*, 2010.

[7] J. O. Kephart, "Research challenges of autonomic computing," in *Proceedings of the 27th international conference on Software engineering*. ACM, 2005, pp. 15–22.

[8] J. Q. Ouyang, S. Dian Xi, D. Bo, F. Jin, and H. M. Wang, "Policy based self-adaptive scheme in pervasive computing," *Wireless Sensor Network*, vol. 1, no. 01, p. 48, 2009.

[9] W. F. Truszkowski, M. G. Hinchey, J. L. Rash, and C. A. Rouff, "Autonomous and autonomic systems: A paradigm for future space exploration missions," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 36, no. 3, pp. 279–291, 2006.

[10] B. Meehan, G. Prasad, and T. M. McGinnity, "A framework for autonomic software," in *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*. IEEE, 2007, pp. 777–782.

[11] P. Romano, R. Giugno, and A. Pulvirenti, "Tools and collaborative environments for bioinformatics research," *Briefings in bioinformatics*, vol. 12, no. 6, pp. 549–561, 2011.

[12] R. T. da Silva, J. M. de Souza, and J. Oliveira, "Autonomic analysis of social networks," in *Computer Supported Cooperative Work in Design (CSCWD), 2011 15th International Conference on*. IEEE, 2011, pp. 508–515.

[13] Z. Yu and X. Zhou, "Socially aware computing: Concepts, technologies, and practices," in *Mobile Social Networking*. Springer, 2014, pp. 9–23.

[14] A. Begel and R. DeLine, "Codebook: Social networking over code," in *Software Engineering-Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*. IEEE, 2009, pp. 263–266.

[15] E. Mainsah, "Automatic computing: the next era of computing," *Electronics & Communication Engineering Journal*, vol. 14, no. 1, pp. 2–3, 2002.

[16] H. Kang, X. Li, and P. J. Moran, "Autonomic sensor networks: A new paradigm for collaborative information processing," in *Dependable, Autonomic and Secure Computing, 2nd IEEE International Symposium on*. IEEE, 2006, pp. 258–268.

[17] N. Samaan and A. Karmouch, "Towards autonomic network management: an analysis of current and future research directions," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 3, 2009.

[18] "Galatean Risk and Care Environment, GRaCE," www.egrist.org, 2017, accessed July 21st.

[19] C. D. Buckingham, A. Ahmed, and A. Adams, "Designing multiple user perspectives and functionality for clinical decision support systems," in *Proceedings of the 2013 Federated Conference on Computer Science and Information Systems*, M. Ganzha, L. Maciaszek, and M. Paprzycki, Eds. IEEE, 2013, pp. 211–218, catalogue number CFP1385N-ART.

[20] C. D. Buckingham, "Psychological cue use and implications for a clinical decision support system," *Medical Informatics and the Internet in Medicine*, vol. 27, no. 4, pp. 237–251, 2002.

[21] Y. Xiao, "Artifacts and collaborative work in healthcare: methodological, theoretical, and technological implications of the tangible," *Journal of biomedical informatics*, vol. 38, no. 1, pp. 26–33, 2005.

[22] C. Khorakhun and S. N. Bhatti, "Remote health monitoring using online social media systems," in *Wireless and Mobile Networking Conference (WMNC), 2013 6th Joint IFIP*. IEEE, 2013, pp. 1–8.

[23] S. Almomen and D. A. Menascé, "An autonomic computing framework for self-managed emergency departments." in *HEALTHINF*, 2011, pp. 52–60.

[24] L. Lapointe, J. Ramaprasad, and I. Vedel, "Creating health awareness: a social media enabled collaboration," *Health and Technology*, vol. 4, no. 1, pp. 43–57, 2014.

[25] B. Long, "Autonomic computing," *Available at:)(Accessed April 13, 2009) http://kbs. cs. tuberlin. de/teaching/sose2006/oc/folien/AutonomicComputingPaper. pdf View in Article*, 2009.

[26] A. Computing *et al.*, "An architectural blueprint for autonomic computing," *IBM White Paper*, vol. 31, 2006.

[27] A. G. Ganek and T. A. Corbi, "The dawning of the autonomic computing era," *IBM systems Journal*, vol. 42, no. 1, pp. 5–18, 2003.

[28] N. Hussain and H. H. Wang, "Semantic enabled social-collaborative research framework for proteomics domain," 2015.

[29] F. Abbattista, F. Calefato, D. Gendarmi, and F. Lanubile, "Incorporating social software into distributed agile development environments," in *Proceedings of the 23rd IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press, 2008, pp. II–46.

[30] C. D. Buckingham, A. Adams, L. Vail, A. Kumar, A. Ahmed, A. Whelan, and E. Karasouli, "Integrating service user and practitioner expertise within a web-based system for collaborative mental-health risk and safety management," *Patient education and counseling*, vol. 98, no. 10, pp. 1189–1196, 2015.

[31] G. Gauvrit, E. Daubert, and F. Andre, "Safdis: A framework to bring self-adaptability to service-based distributed applications," in *Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference on*. IEEE, 2010, pp. 211–218.

[32] H. Harroud and A. Karmouch, "A policy based context-aware agent framework to support users mobility," in *Telecommunications, 2005. advanced industrial conference on telecommunications/service assurance with partial and intermittent resources conference/e-learning on telecommunications workshop. aict/sapir/elete 2005. proceedings*. IEEE, 2005, pp. 177–182.

[33] T. M. King, A. E. Ramirez, R. Cruz, and P. J. Clarke, "An integrated self-testing framework for autonomic computing systems." *JCP*, vol. 2, no. 9, pp. 37–49, 2007.

[34] J. McCann and M. Huebscher, "Evaluation issues in autonomic computing," in *Grid and Cooperative Computing-GCC 2004 Workshops*. Springer, 2004, pp. 597–608.