Scalable Computing:
Practice and Experience

Scientific International Journal
for Parallel and Distributed Computing

ISSN: 1895-1767

Universitatea de Vest din Timişoara

Volume 14(3) September 2013

SUBSCRIPTION INFORMATION: please visit http://www.scpe.org

Scalable Computing: Practice and Experience

Volume 14, Number 3, September 2013

TABLE OF CONTENTS

DOI 10.12694/scpe.v14i3.855
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 14, Number 3, pp. iii–iv. http://www.scpe.org

INTRODUCTION TO THE SPECIAL ISSUE ON CONTEXT-AWARE ARCHITECTURES AND SERVICES ON EMERGING INFRASTRUCTURES

Dear SCPE readers,

Future information society is full of exciting and new applications and services, may it be a smart transportation system, an ambient living environment, or large-scale sensor networks for climate prediction. These services, among other requirements, rely on context-awareness (e.g. user behaviors, user intentions, the environmental status/ambiance, etc.). Consequent systems can adapt themselves through resource management or reconfiguration to achieve specific goals, such as functions, performance, energy budget and reliability. This area of research is challenging yet instrumental for the incoming digital era.

The unique articles submitted to the SCPE special issue on Context-aware Architectures and Services on Emerging Infrastructures" have undergone a rigorous peer-review process. The manuscripts were reviewed by at least two independent reviewers each and the guest editors finally performed meta-reviews on the papers, and according to an objective score sheet, the 2 best rated articles (out of 6 submissions; acceptance rate 33%) where finally selected for publication.In conclusion, we greatly appreciate all the authors and reviewers for their contribution to shaping this special issue.

The two papers accepted in this special issue present very interesting insights, surprisingly, both from the business and industrial perspective. The first article "Context-aware services in cooperative value chains: a key player-centered approach" attempts to promote the adoption of ubiquitous context-aware applications through exploiting the tangible benefits of key players/stakeholders. It presents a framework to identify the key players, their individual concerns and benefits, and to integrate these considerations in the deployment of a context-aware service. Wireless monitoring of valuable items in the insurance sector is demonstrated as a case study. The other article "ELL-i: An inexpensive platform for fixed things" is an invited contribution from a startup company. ELL-i is an open-source (in terms of both software and hardware) platform for Internet-of-Things applications with Power-over-Ethernet (PoE). In contrast to the mainstream wireless connection approach, ELL-i innovates on adaptive monitoring and control via wired connections. The co-operative type ownership model is intended to gather the widest possible joint efforts to develop services and products based on the open-source platform.

Ethiopia Nigussie, *University of Turku, Finland*
Andreas Riener, *Johannes Kepler University Linz, Austria*
Liang Guang, *University of Turku, Finland*

DOI 10.12694/scpe.v14i3.856
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 14, Number 3, pp. 139–153. http://www.scpe.org

CONTEXT-AWARE SERVICES IN COOPERATIVE VALUE CHAINS: A KEY PLAYER-CENTRED APPROACH

CHRISTINE BAUER[†], CHRISTINE STRAUSS[‡], CHRISTIAN STUMMER[§], AND ALEXANDER TRIEB[¶]

Abstract. The progress in context-aware systems is predominantly technology-driven, leading to innovations in technology-savvy industries while having almost no impact on other industries, such as the insurance sector. In this paper we introduce a novel, key player-oriented framework that explicitly takes into account the (business) value for each player being relevant when implementing a particular context-aware service on the market. We illustrate the framework's applicability by means of a case study of a context-aware system that allows for monitoring high-value items for insurance purposes.

Key words: context-aware service, key player, value-orientation, business value, business approach, insurance sector, high-value items monitoring

AMS subject classifications. 93A02, 94A02

1. Introduction. Advanced hardware miniaturisation, the continuous increase of capabilities for communication, processing, and storage, as well as the proliferation of electronic devices are key drivers towards context-aware computing applications. Although context-aware technologies are becoming more reliable and cost efficient and, thus, businesses are exploring ways in which they can leverage them for novel services in order to improve their market position [4], there is still no consensus on what they should be used for [21] and how to realise the vision of the internet of things [17].

Innovation in the field of context-aware systems is mostly technology-oriented, in fact even technology-driven [21]. Following this approach, the innovation potential has started to (slowly) emerge within traditional manufacturing as well as in distribution and retail industries [9], but so far had no or just negligible impact on other industries. As we will show in our case study from the insurance sector, the reason for the hampered adoption of context-aware computing neither lies in the non-applicability of context-aware computing nor in the lack of creativity to come up with new ideas. Rather, many industries are driven by business value considerations instead of being predominantly pushed by technology. As a result, benefits expected from an investment in a new business idea have to clearly exceed the anticipated risks. To this end, Fleisch and Tellkamp [8] proposed a framework aiming at identifying value-creating applications in the ubiquitous computing domain, focusing on the "challenges identifying value-creating applications both ex ante and ex post". This framework later on was extended by Strauss et al. [27], who introduced a decision support guide for the evaluation of the value of an envisaged service that may be used by businesses to decide based on quantitative criteria whether or not to invest in a project and offer a novel service. They also provided a case study on damage prevention, where they analysed the function of several market players from different fields in offering a context-aware computing solution. They show that each market player fulfils a crucial role in realising the particular service.

Such two-sided or multi-sided markets, in which two or more distinct user groups benefit from each other by interacting on the market [18], can be found in several industries (e.g., health sector, telecommunications, gaming industry, etc.). The market for credit card services, for instance, comprises cardholders, financial institutions, merchants with a wide-spread network, and developers of credit card readers; all these market players make up a market by cooperating and establishing mutual trust. Offering context-aware services also typically relies on several market players, i.e., hardware suppliers who provide the technological infrastructure, service providers who use this infrastructure to provide their services, and (end-)users consuming the service [2]. Since a service cannot be realised in the absence of any of these stakeholders, benefits and risks of each key player have to be

[†]Vienna University of Economics and Business, Department of Information Systems and Operations, Welthandelsplatz 1, 1020 Vienna, Austria (chris.bauer@wu.ac.at).
[‡]University of Vienna, Faculty of Business, Economics and Statistics, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria (christine.strauss@univie.ac.at).
[§]Bielefeld University, Department of Business Administration and Economics, Universitätsstraße 25, 33615 Bielefeld, Germany (christian.stummer@uni-bielefeld.de).
[¶]travel audience GmbH, Mozartstraße 4b, 53115 Bonn, Germany (a.trieb@travelaudience.com).

balanced such that the envisaged service can be realised.

Two alternative scenarios for market development have been observed in practice. The first one is driven by a cooperative network [13] that is formed in five phases (i.e., initiation, forming, storming, norming, performing [28]) during which potential business partners get to know each other, build mutual trust, and set joint goals. Then they jointly elaborate a business idea, for which each potential network partner individually evaluates its own benefits, costs, and risks. In the second scenario, one entity (i.e., an organisation or an entrepreneur) propels the business idea and strives to 'sell' it to other businesses that are required to operationalise the idea. In this case, we assume that potential partners may be convinced more easily if expected benefits are evident. This constitutes the starting point for our research: Given business-value as major driver as well as multi-sided markets, the paper at hand proposes a novel procedural framework that uses a 'key player-centred approach' to identify valuable context-aware services.

The remainder of this work proceeds as follows: Firstly, we discuss existing approaches and related work (Sect. 2). Next, we introduce our novel framework that puts the key players' benefits into the centre (Sect. 3). Then, we show the applicability of the framework by means of an exemplary case study, i.e., the context-aware service for monitoring high-value items in the insurance industry. This case study demonstrates the framework's value in leveraging context-aware computing technologies to gain benefits for both insurance companies and their customers (Sect. 4). The final section summarises the research contributions of our work (Sect. 5).

2. Related Work. In this section, we present selected research on evaluation and valuation; this section is the foundation of and justification for the framework, which we present in Sect. 3.

In an attempt to add a perspective that differs from the (usual) orientation on technological issues, Oulasvirta [21] postulated that any technology should be leveraged for the service on humanity and, therefore, research needs to be directed towards the technology's relevance to people, the understanding of people and their activities, and the empowerment of people themselves to address their needs for design goals. This approach borrows from user-centered (product concept) design [16, 20], which is widely adopted in the field of human-computer interaction (HCI). However, putting user needs into the loop is not sufficient to make a business idea economically feasible.

Social computing [25] deals with embedding technology in a social context and emphasises that social change, that has been caused by technology, has to be considered as well. Analogously, the value-sensitive design approach by Friedman [10] places emphasis on the role of human values and morals in deciding what features or characteristics of technology are relevant and are particularly worth of being further pursued in system design (e.g., user autonomy). This approach, however, does not take economic feasibility into account. Furseth, Cuthbertson, and Reynolds [11], in contrast, argue that service innovation is "driven by the value created for the customer, where value may be defined in both economic and social terms". While this point of view imbeds the customer's value, it still remains one-sided because this value does not automatically translate into willingness to pay for a service (which may be particularly true in a "for free-economy" setting) or leads to (other) revenues that may at least even out the costs. Therefore, the critical question remains how an entity (e.g., a company) should be motivated to offer a service that does not promise a positive, direct or indirect, value for the provider.

The framework by Fleisch and Tellkamp [8] addresses this issue by analysing the value for each participating entity (with respect to both tangible and intangible benefits). As shown in Fig. 2.1, potential sources of value can only yield benefits for the involved entities when several challenges, referring to the network, constraints, implementation issues, and valuation, have been overcome. Analysing network effects is necessary in order to get a clear(er) picture of the potential sources of value. Constraint challenges may inhibit that the entire project can be realised. Implementation challenges refer to limited resources and knowledge in the implementation phase or during actual use of the service. While network, constraint, and implementation challenges diminish the value of a project or hinder its realisation, valuation challenges are problems of determining the tangible benefits from a service either ex ante or ex post.

For their decision support framework, Strauss et al. [27] adopt the above concept, and enrich it further with a scenario technique to allow a decision whether or not to invest in an innovation based on feasibility and efficiency criteria. Although cost forecasts are no explicit part of their framework, they build the basis for a (quantitative) assessment based on different prototypical scenarios. Their framework suggests to have a

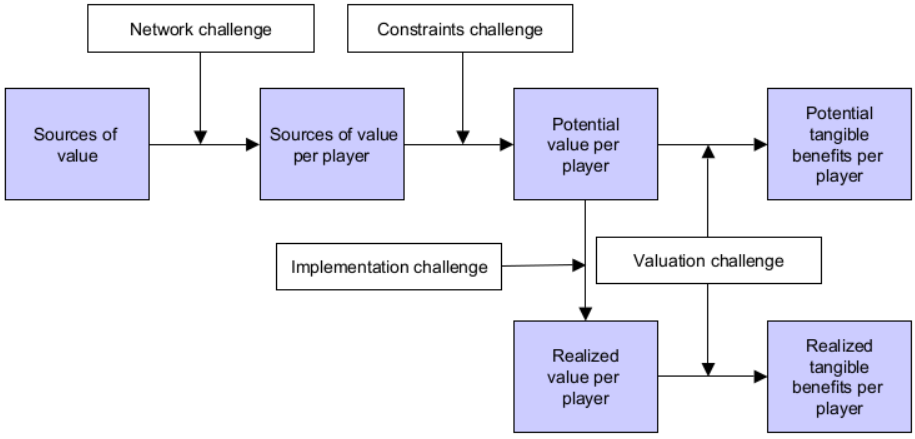

FIG. 2.1. *A framework for identifying value-creating applications in ubiquitous computing [8]*

much closer look at the costs and tangible benefits (by including pessimistic and optimistic estimates to provide a spectrum in costs and relative values of tangible benefits) and to develop both aggressive and conservative scenarios to be able to identify differences in customer-related drivers (e.g., number of sensors per site). Thus, a more reliable basis for the decision on whether or not to invest in an innovative service can be established.

Pang et al. [22], finally, claim a 'business-technology joint design' intended to cross the gap between industrial business practices and technology development. Both, technical and business challenges, have to be resolved simultaneously to bring successful context-aware solutions into the market. They emphasise that value creation on the business side is essential, and information and technical requirements have to build on and towards this business value. Further, they argue that a value-centric approach is superior to a technology-centric one as users will pay for values, not for technologies. In following such a value-centric approach, the value creation should be related to the entire set of stakeholders of the envisaged service.

3. The Key Player-centred Framework. Developing a framework is an inherently iterative search and design process that follows continuous cycles of generating and evaluating design alternatives. Accordingly, we gradually improved the framework and used group reflection phases after each generation/evaluation cycle to foster them towards the final coherent framework. For evaluation purposes we chose an ex post approach (cf. the strategic evaluation framework of Pries-Heje et al. [23]). The work of Venable et al. [29] proved as a useful guide in choosing an appropriate method. Out of the seven methods that support an ex post approach, we finally opted for the case study approach, because it illustrates not only the benefits very clearly, but also delivers a proof of concept. The case study will be presented in Sect. 4.

For an overview of the procedural framework see Fig. 3.1. It starts with a business idea for a context-aware service in a given domain that recombines resources in a way that allows for pursuing a business opportunity [26]. Typically, this includes selling a product or a service, although alternative approaches exist as well (e.g., business models that offer free services to users and are financed through advertising).

The next step is the identification of relevant key players, who have to commit to a project, such that the idea can be realised. While some projects may be realised by only one key player, most business ideas including context-awareness require the cooperation of several players (e.g., [3, 5]).

In a commercial setting, each of the key players has to get the opportunity to gain some (individual) value from the envisaged service (otherwise, they would not contribute). The third step in the process is therefore dedicated to identifying and determining the benefits for each key player. These benefits may be tangible (e.g., profit, reduced employee turnover, reduced costs, etc.) or intangible (e.g., improved employee morale, heightened customer loyalty, better vendor relationships, image improvement, etc.).

After having successfully determined benefits for each of the key players, scenarios for the envisaged service need to be developed. For three reasons the development of scenarios has to be performed independently of any

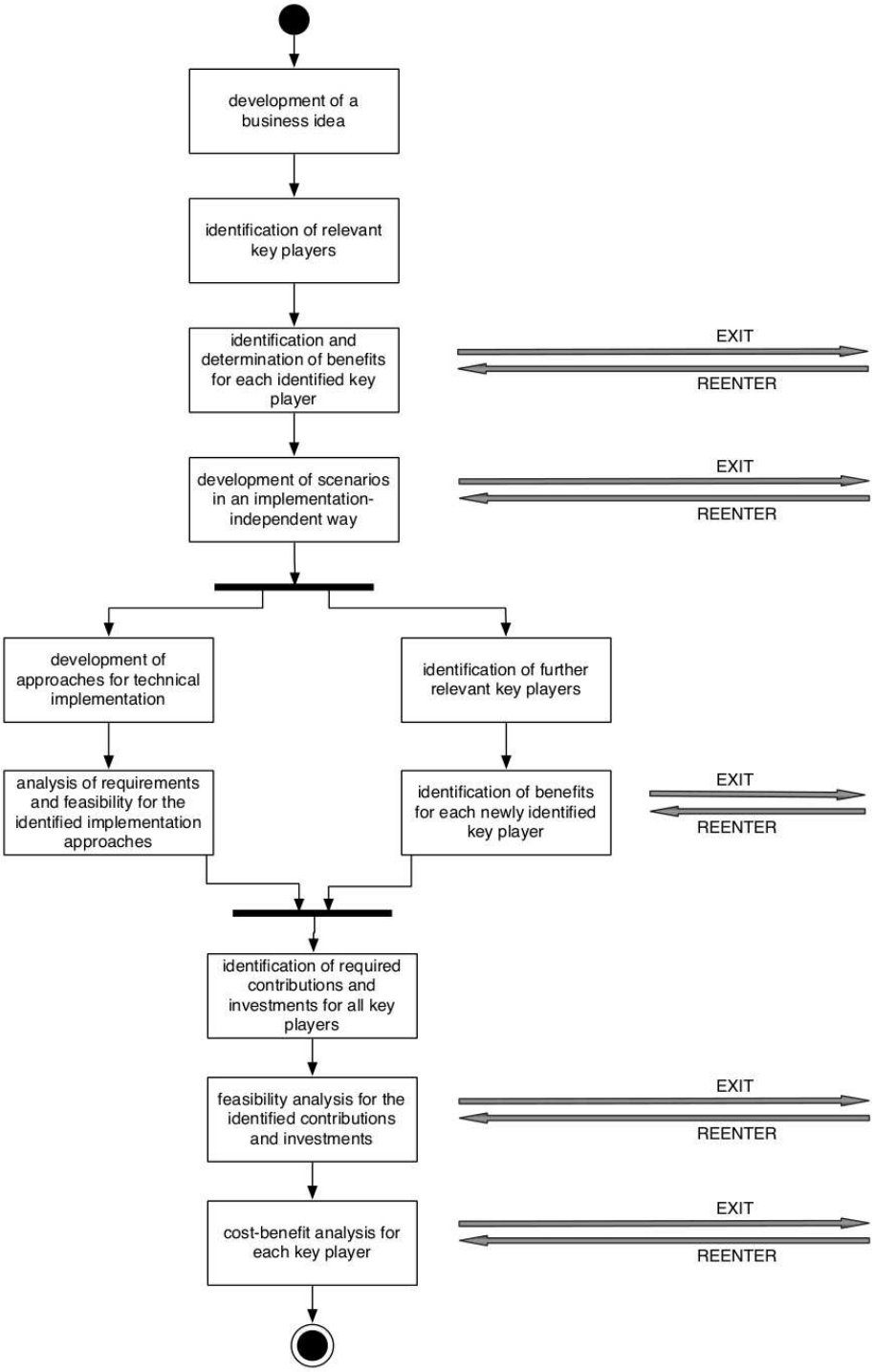

FIG. 3.1. *The key player-centred approach*

(possible) technical implementation: First, the scenarios should be defined by managers, while the technical implementation possibilities will be identified by people with technical background (e.g., computer scientists, engineers), who come into play at a later stage. Second, services envisioned may not be technically feasible in a certain point in time, but later on they might. As a result, it will be easier to come back to the scenarios later when it is described in an implementation-independent way, whereas any implementation-driven approach would require to start over in elaborating the scenarios. Third, for context-aware services, context is the pivotal element. Similar to the definition of functionality of the envisaged service, context should be defined as required from the business perspective because, from a technological perspective, sourcing context can be performed in various ways. For instance, Dobson [6] presents a non-exhaustive taxonomy of 18 different implementation approaches to gather location context information, in many cases derived from other sources than from explicit location-sensing hardware at all.

After the development of scenarios, technical implementations for the scenarios have to be identified. These implementation alternatives should then be analysed for requirements and feasibility. This will include considering the pros and cons of the various implementation approaches. As technical experts are usually aware of what is currently possible from a technical perspective, we expect that the two steps of identifying implementation alternatives and evaluating them might coincide in most cases.

While engaging with technical issues, a second, parallel thread in the process takes again a business perspective. Based on the developed scenarios, it should be verified if the key players involved so far will be able to cover all aspects described in the scenarios, or if additional players need to be considered to realise the new service project. In the latter case, benefits need to be identified also for the newly identified key players.

After having successfully addressed both, the technical as well as the managerial thread, the process continues with the identification of required contributions and investments of each key player. Such contributions and investments may be of various nature including process adaptation, distribution, marketing efforts, financial contribution, provision of space, legal advice, technology, or networks and so forth. Having identified the required contributions, another analysis concerns the feasibility and prospective willingness of key players to contribute as required. Finally, a cost-benefit analysis for the envisaged service has to be performed separately for each key player involved.

As Fig. 3.1 indicates, there are several exit points throughout the process that allow to abandon the project, either completely (e.g., if only insufficient benefits can be identified) or with the intent to resume the project at a later point in time (e.g., if technologies have to be more advanced before a service can be realised in the required way). Therefore, each exit point is at the same time a possible "point of return" to reenter the project, when, for instance, technology is more advanced, additional benefits have been generated for one of the key players, etc.

4. A Case Study in the Insurance Sector for Monitoring High-Value Items. In the following, we describe a case study from the insurance industry, in which a context-aware infrastructure is leveraged to gain benefits for insurance companies, their customers, and other key players involved. The presented case is based upon the premise of providing anxiety-relieving monitoring and risk services by being always connected, always aware, and demonstrating pro-activity with the customer.

4.1. Business Idea: Wireless Monitoring of High-Value Items. The insurance industry is confronted with an increasing number of theft incidents. A context-aware infrastructure could offer a solution to minimise the issues of theft or damages particularly for high-valued insured items (e.g., paintings, sculptures, furniture, jewellery, tapestries, antiques). Many of these valuable items are typically not placed in a high-security location and may for this reason be targeted in a burglary. Besides the monetary value, the emotional value of such items to the owner should not be underestimated. Therefore, customer loyalty may be substantially increased, once a solution can be provided that improves the likelihood of recovering stolen items.

The principle business idea is to provide a context-aware service that allows for monitoring the current state of high-value items, including tracking their location and environment conditions. In case of a critical event, the item's owner (i.e., the insured customer) as well as the insurance company are instantly notified, and stolen items are automatically registered at dedicated institutions for art theft.

4.2. Identification of Relevant Key Players and their Benefits. The primary stakeholders of the envisioned context-aware service include the insurance company and the insurance customer (i.e., the owners). Within the insurance company, the users of a system that processes data for this service will include the insurance agent, insurance appraiser, and customer service representatives. Granting professional art historians and dealers access to the system would further streamline the evaluation process of high-valued items. The tangible and intangible benefits identified from the perspective of the insurance company are summarised in Tab. 4.1.

TABLE 4.1
Benefits for insurance companies

Tangible Benefits	Intangible Benefits
Reduction in Claims Expenses	Increased Customer Loyalty
Reduction in Administrative Overhead	Increased Brand Awareness
Reduction in Customer Visits	Improved Customer Retention
Increase in Premium Revenues	Improved Efficiency of Resources
	Reduction in Theft Risk

Reduction in claims expenses would offer the most obvious tangible benefit for the insurance company due to the higher probability of recovering stolen items and the minimisation of damages requiring substantial repair and restoration expenses. Moreover, even the administration overhead for claims could be reduced by automatically initiating the pre-filling of the claim with the required information, such as items missing, time of day, location, insurance policy holder, or total projected claims value. Cost savings in administration may be relatively low compared to the other expenses mentioned before; however, this feature would demonstrate pro-activity to the customer, and thus may contribute to positioning the insurance company as a trustworthy helper and problem solver.

Moreover, being the first insurance company to offer such a service, the corresponding first mover advantage may yield an increase in overall premiums as a result of a potential rise in market share attested to growing interest and demand of the innovative solution. Also, customer visits from insurance appraisers and other professional art historians could be reduced when online communication capabilities are realised through the micro-portal, i.e., a network and platform on which art dealers, appraisers, customers and agents can interact. Furthermore, a potential network effect could arise through the instalment across the entire customer base, which may contribute to a general decline in theft risk, which, again, may translate into lower premium needs that may be passed on to the customers as an additional incentive.

The tangible and intangible benefits identified from the perspective of the insured customer are summarised in Tab. 4.2. Overall, the envisaged service may contribute to resolving the insurance companies' dilemma between the customers' expectation of strong commitment to safety and security of their high-valued items on the one hand, and an increasing price-sensitivity on the other hand.

TABLE 4.2
Benefits for insured customers

Tangible Benefits	Intangible Benefits
Reduction of Premiums	High Safety and Security
Reduction in Administrative Efforts	Increased Emotional Value
Reduced Damages	Increased Recovery Probability of Stolen Items
	Increased Improved Service Quality

4.3. Scenario and Functionality: Wireless Monitoring of High-Value Items. Based on the principle business idea (cf. Sect. 4.1), a set of scenarios were elaborated and described in detail. Based on these scenarios, various functions for a context-aware monitoring service provided by an insurance company were derived and listed in Tab. 4.3. This list includes inventory management functionality that allows for adding and deleting items and for updating item information. Further, information from all insured items at each location

is aggregated and, thus, available for analysis purposes. Alert messages from a base station are constantly monitored; in case of an alert, customers are automatically notified via short messaging (SMS), e-mail, and/or through an insurance representative (e.g., broker, agent, or service representative), and the claims process is automatically triggered. Stolen items are identified in real-time, and potential claim damages are immediately assessed. Moreover, stolen items are automatically registered at two major web services for registering stolen art: Interpol's Recent Thefts Database [15] as well as at Art Loss Registry [1].

The probably most appealing functionality is the monitoring of items at any time via a web interface. This feature is particularly important to monitor items in transit for additional protection and to track stolen items to locate and recover them within a short timeframe. Environmental conditions may also be monitored which allows for fast notification of potential damages and quick response in order to minimise the extent of such damages. Optimal temperature and humidity conditions are essential, for instance, for paintings. Furthermore, historical data on environmental conditions are collected to reveal new risks or long-term effects.

Table 4.3
Functions of the context-aware monitoring service

Functions
Inventory management
Aggregation of information from all insured items
Monitoring of alert messages
Automatic notification of critical events to the customer
Real-time identification of stolen items
Real-time assessment of potential claim damages
Automatic registration of stolen items at Interpols Recent Thefts Database
Automatic registration of stolen items at the Art Loss Registry
Online tracking of items
Monitoring of items in transit
Historical data of environment conditions

Via a customer web portal (for functionalities cf. Tab 4.4), each customer has an up-to-date record on the entire inventory of his or her insured items, which contains a description, photographs, and information about the value of each item. An inventory management component enables to add and delete items and to update information on insured items. This component further manages the insurance policy and the premium payments. When new items are added, its documentation is automatically verified by notifying an insurance appraiser or professional during the operation. Further, a customer can access information of insured items' environmental conditions via the web portal at any time. Items in transit may be monitored to constantly be informed of their locations. In addition, the portal provides a direct communication channel to an insurance representative (i.e., agent, broker, customer service representative, etc.).

Table 4.4
Functions of the customer web portal

Functions
Presentation of the entire inventory of insured items
Online inventory management
Automated adding and documenting of the inventory
Automatic verification of the documentation of a new item
Presentation of environmental conditions
Monitoring of items in transit
Direct communication channel to an insurance representative

Overall, the envisioned context-aware service allows for identifying each individual insured item, providing sensing information of environmental conditions, automating alert messaging, responding pro-actively to critical events, and increasing recovery probability of stolen items.

4.4. Technical Implementation and Feasibility Analysis. The envisaged context-aware system is composed of sub-systems for data gathering, data communication, data analysis and monitoring, as well as item tracking and recovery processes. The data gathering process is responsible for capturing environmental conditions as well as security events. The latter may be the movement of an item or the vibration of a painting that is being cut out of its frame. These events are captured by a central reader that filters the information and forwards the data to the backend context-aware insurance system via the cellular network. Figure 4.1 provides an overview of the technical architecture of the proposed context-aware service solution.

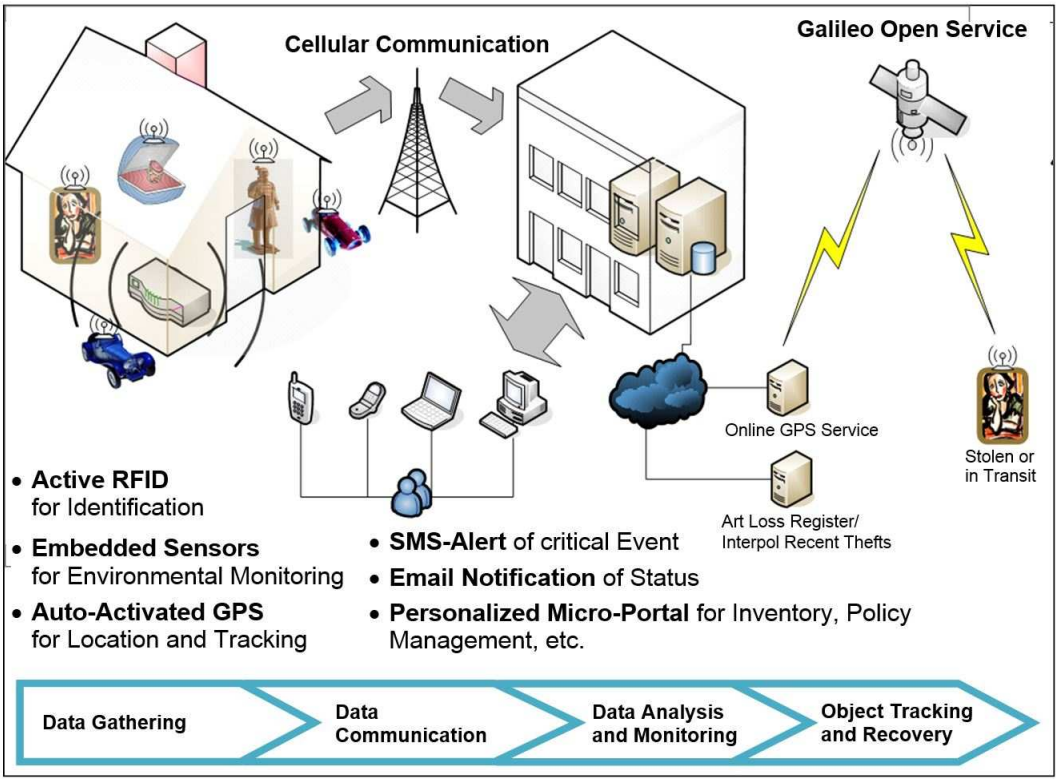

FIG. 4.1. *Technical architecture for the high-value items tracking solution*

The individual tags are comprised of Class 3 semi-passive Radio Frequency Identification (RFID) tags with integrated humidity, temperature, and vibration sensors including an embedded Global Positioning System (GPS) and General Packet Radio Service (GPRS) communication module. Figure 4.2 summarises the proposed component architecture of the tag. Class 3 semi-passive RFID tags were chosen over alternative tag types due to their lower monetary costs (compared to active RFID tags), read/write capabilities, integrated sensors, extended signal range, and built-in security. The tags exhibit additional benefits such as maximisation of the life span of the power source due to the so-called 'deep sleep' mode as well as the backward compatibility to standard passive tags [7]. The 'deep sleep' mode allows tags to use battery when signals are received from the reader. When in 'deep sleep' mode, the tag reverts to the standard inductive method found in standard passive tags. Through this capability, the tags could have an expected sensing lifespan of over five years with transmission intervals of every 20 seconds and an unlimited identification lifespan.

The tags also include GPS with an integrated GPRS communication module to enable real-time tracking during the event of a theft. Due to the high power requirements of GPS modules, they would remain in a non-operational state until the RFID signal gets lost or vibrations are sensed; when this happens, it can be assumed that the high-valued item is being stolen. If so, the GPS module switches into location and tracking mode and GPRS communication is used to send the GPS coordinates to an online server which allows external companies, such as insurance companies, to retrieve the tracking information via a standard web service interface. The high

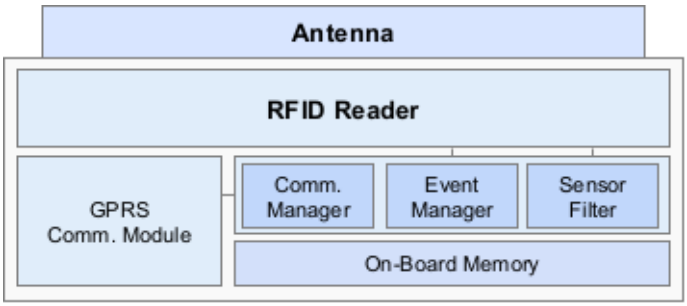

FIG. 4.2. *Customised RFID tag*

power consumption of the GPS and GPRS modules shortens the further lifespan of a tag to a few months. It is assumed, however, that the time required to locate and retrieve a stolen item will be accomplished within this time frame. Note, that we suggest to add temperature and humidity sensors in order to provide an additional advisory service: tracking humidity and temperature conditions for an insured item enables an alert at the base station once the measured values either exceed or fall below some predefined (acceptable) threshold values. This service could prevent or at least minimise additional damage.

The proposed solution requires the installation of a base station, which incorporates an RFID reader, software components, and a cellular communication module. Figure 4.3 provides an overview of the technical components of the base station. The RFID reader aggregates information collected from the tagged items. The software component manages the aggregation, filtering, and exchange of information. The event manager listens to the sensor filter component and and alerts the context-aware insurance system, if temperature or humidity either exceed or fall below some predefined threshold values or the vibration sensor has been triggered. Events can also be timed-based in order for historical data, which can be stored in the on-board memory, to be sent to the context-aware insurance system on a daily, weekly, or monthly basis. The connection is established through the GPRS cellular network; a standard Internet Protocol (IP) connection is used for the communication between the backend servers and the reader, which in essence is a traditional client-server connection.

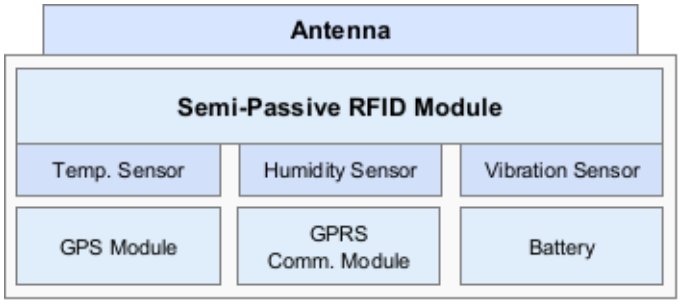

FIG. 4.3. *Base station architecture*

The data analysis and monitoring process is characterised by the exchange, storage, analysis, and presentation of data. Due to the IP based connection between the base station and the backend system, an existing middleware, workflow management, and database infrastructure for message passing, event triggering, and data storage could be used. Both the insurance company and the customer can access the information through various web-enabled devices such as mobile devices, tablets, and laptops.

Finally, a feasibility analysis has to be performed for this envisaged implementation in terms of technology as well as in terms of limited resources.

4.5. Identification of Additional Key Players and their Benefits. In the course of describing the envisioned context-aware service and its functionality in more detail, Interpol and the Art Loss Registry have been identified as additional key players. Currently they run the Recent Thefts Database [15] and the Art Loss

Registry [1] as web services that allow for registering stolen art and provide an opportunity for art dealers to check whether an offered item has been reported stolen. If these two players would limit or close their services, the benefits of the envisioned context-aware service would decrease. The benefits for these two players have been summarised in Tab. 4.5.

TABLE 4.5
Benefits for Interpol and the Art Loss Registry

Tangible Benefits	Intangible Benefits
Reduction in Administrative Overhead	Improved Efficiency of Resources
Increased Usage of their Databases	Increased Quality of Service

Also, an online tracking provider is essential for the context-aware service; an appropriate candidate needs to be found and integrated. Similar to Interpol and the Art Loss Registry, the online tracking provider would benefit from the service through an increased demand of its specific services, which also may have a positive impact on the image of this provider.

Another expansion could include police departments and/or a security company. This would allow for an automatic notification of stolen items as well as for the exchange of information regarding the description (including photographs) and the location, which may help to speed up the recovery of stolen items. Thus, the success rate in the recovery of stolen items may increase.

Given that insurance products and related services may be complementary to existing products and/or services, a partnership between an insurance company and system integrators or security providers could be worth to be considered. The context-aware service would then only be a specific functionality offered with an existing security system. Still, the monitoring information should be made available through the insurance company in order to establish and maintain a constant communication channel between insurer and customer. For the further analysis in this case study, however, we stick with the original idea that an insurance company offers the context-aware service.

4.6. Identification of Required Contributions and Investments. After having gathered the primary functions and designed the technological architecture for the context-aware service, this section elicits contributions and investments necessary for its realisation.

Insurance Company. The realisation of the technical architecture would require an initial investment for the customised development of the underlying monitoring system, the base station software (assuming that software based on these requirements does not exist), and the installation. For running the service, further fixed as well as variable costs have to be added.

The initial development efforts for the backend system depends largely on the existing system infrastructure of the participating insurance company. As existing systems within the insurance industry are heterogeneous, fragmented, and complex, it is very likely that the participating insurance company would have to invest in a new (service-oriented) system in order to allow for an exchange between the context-aware insurance system and existing systems (such as customer relationship management, workflow management, or databases) with limited subsequent maintenance requirements.

Alternatively, a solution in which the insurance company does not change its technical infrastructure to a service-oriented structure could be feasible. However, it may become difficult to ensure the exchange of information between the context-aware system and existing systems. In addition, customised development for the integration of the new system would be required, which may exceed reasonable investment and may also subsequently involve high maintenance efforts. Independent from the decision between going service-oriented or maintaining the basic infrastructure as is, skilled staff has to be available for the development of the base station and the backend system. Whether realised with in-house capacities of the insurance company or being outsourced to an information technology (IT) company, the insurance company has to bear the costs for the customised development.

Furthermore, it has to be considered that the envisaged context-aware service could imply competition with companies outside the insurance industry because, in recent years, developments in security solutions for

works of art have emerged on the market, offering theft prevention and asset tracking solutions. However, these solutions were developed for museums and similar institutions with a security infrastructure and security personnel. Therefore, these are doing well by being based on active ultra-long-range RFID technologies, which only allow for tracking within buildings that are equipped with a complex RFID architecture. Consequently, these solutions cannot be compared directly to the proposed solution.

Finally, valuation challenges exist in estimating the system's benefits prior to its actual implementation. The challenges for determining the tangible benefits of the context-aware system ex ante relate to the inability to identify the probability for recovering stolen items, the reduction in damages due to environmental conditions as well as determining the potential increase in customer loyalty and brand awareness that is hoped for as a result from such a product offering. Measuring the value of the context-aware service after it has been implemented should be rather straightforward since the recovery rate would be known, which allows for determining the reduction in settlement payments and, to some degree, the increase of customer loyalty.

Insured Customers. From the perspective of the owners of high-value items, a basic requirement for the success of the context-aware system lies in their willingness to have their high-value items tagged which, however, may constitute a major barrier to them. From a financial perspective, though, the investment in tags and base station are quite low; still, it has to be decided whether the customer or the insurance company will pay for them (or how costs should be split).

Online Tracking Provider, Art Loss Registry, and Interpol. The online tracking provider, the Art Loss Registry, and Interpol do not have to directly contribute financially, but they need to provide their services and show some commitment to keep them alive. In case of a cooperation with police departments and/or a security company, these institutions may have to upgrade their systems and use recent RFID-, camera- and Internet-enabled mobile devices.

4.7. Feasibility Analysis. Next, the feasibility with respect to required contributions and investments for the realisation of the context-aware service will be investigated.

Insurance Company. An insurance company has to invest in reshaping its existing systems in order to achieve a smooth integration with the envisaged context-aware service. Under the assumption that it will be an insurance company that ultimately will drive the implementation of the business idea, the necessary commitment can be expected, because anticipated benefits are higher than projected costs and risks. Furthermore, insurance companies are under pressure to develop effective measures for controlling claim costs.

With respect to implementation, several challenges need to be overcome. The miniaturisation of GPS modules is technically feasible, although price may be an issue. Further, security concerns exist with respect to RFID; guaranteeing RFID security, thus, remains a challenge [19]. The availability of skilled staff for the development of the base station and the backend system can also cause problems in the realisation of the context-aware service. Then, assisted-GPS technology [24] has been introduced to enable indoor location capabilities, but these do not guarantee availability or accuracy of the coordinates. The issue could be resolved through the combined positioning capabilities of the Galileo Positioning System and more advanced cellular networks. An indoor positioning system (IPS) would be an option, but it does not comply with the international standard ISO/IEC 24730 on real-time locating systems (RTLS). Moreover, the validation of the required RFID transmission range within different building types constitutes another challenge. This will be particularly difficult in historic buildings due to the notoriously thick walls. Furthermore, burglar may scan RFID tags using an advanced custom-built RFID antenna which would pose problems with trustworthiness and system security.

Finally, estimations concerning the initial developing efforts for the backend system are tricky without knowledge on the actual system infrastructure at the insurance company. Accordingly, evaluations can only be based on experience and rough estimates. Costs therefore have to be assessed on a case-by-case basis with a thorough understanding of the underlying IT architecture.

Insured Customers. The owner of a high-value item is likely to adopt this novel form of monitoring because of the significant financial value and the high emotional value of the items. However, there could be a psychological barrier for them to apply a tag to such a highly valued item. Basically, there are two options

for installing a tag, either as part of the highly valued item or attached to its container. If installed as part of an item (for instance, drilling a small hole in a statue, placing the tag, and restoring the hole professionally; or sticking a tag non-detachably to the back of a painting), a potential thief could not remove the tag from the item without damaging the item. However, even when the installed tag is invisible, it alters the item, which may cause a devaluation in case of resale. If the tag is attached to an item's container, the item remains in its original condition, but a potential thief may easily withdraw the item from its container, making the precautious tag instalment ineffective. In essence, there is a trade off between 'effective instalment, but compromising the item' versus 'preserving the item, but the instalment may potentially be easily circumvented by a thief'. Compared to the improved likelihood to recover stolen items and to avoid damages by the environment, this barrier should, however, be easy to overcome.

Online Tracking Provider, Art Loss Registry, and Interpol. We assume that the online tracking provider, the Art Loss Registry, and Interpol would participate in this context-aware service, since they do not require any investment, because they have their services ready to provide.

4.8. Cost-benefit Analysis. Costs (as well as their trends) for necessary technical components may be derived from data being available for similar components used in existing asset tracking systems. Cost estimates for the software components used in the base station turn out to be more difficult. Two assumptions regarding their costs have to be made: We assume that the software must be custom-built which requires highly skilled programmers and that GPRS and RFID or sensor filtering software will be available as software components and, thus, require only simple configuration and/or low modification efforts.

Further estimates are required to determine the potential economic valuation of the proposed solution, assuming that a single insurance company will invest into the development and installation of the system. Assuming an equal market share distribution for the top three insurance companies in a country, an approximate claim settlement payments per year can be calculated. Considering an estimated growth rate of, for instance, ten percent for theft incidents and a time frame of five years, the break-even can be estimated. Table 4.6 lists the identified cost drivers including the estimates that are necessary for an evaluation.

Table 4.6
Overview on cost drivers and estimates

Cost Drivers	Estimates
Semi-passive RFID Tags inclusive Sensors	Number of Customers
Base Station (RFID Reader, GPRS Module and Software Component)	Average Number of Tagged Items per Customer
Backend Application Server	Hourly Cost of Installation
Database and Storage Server	Development Cost per Person-day
Software Development (Base Station)	Discount Rate
Software Development (Monitoring)	Theft Growth Rate
Software Development (Frontend)	
Personnel (Installation)	

Strauss et al. [27] recommend scenarios for such an economic evaluation. Following this approach, we developed the four scenarios depicted in Tab. 4.7 for which we distinguish between *pessimistic* and *optimistic* parameters concerning the value of the novel service (e.g., costs and tangible benefits) as well as between *aggressive* and more *conservative* assumptions with respect to the customer base or the number of insured high-value tagged items.

All costs for hardware development and installation have to be included in the evaluation. A difference of roughly fifty percent between pessimistic and optimistic values can be chosen to account for extreme conditions. Software development efforts are estimated assuming that the software is customised, the development can be split for the base station, monitoring, and portal software components, and the fees do not exceed standard low-cost programming fees charged by Eastern European system integrators. Integration efforts have to be excluded at this point, due to the lack of information with respect to existing systems. Assuming a standards-based service-oriented architecture IT infrastructure, the development of the necessary interface and service elements could be considered part of the estimated development efforts. Additional backend hardware components

TABLE 4.7
Systematic scenario development

		Value	
		Pessimistic	Optimistic
Quantity	Conservative	**Scenario I** Costs — high Tangible Benefits — low Customer Base — small Number of Items — few	**Scenario II** Costs — low Tangible Benefits — high Customer Base — small Number of Items — few
	Aggressive	**Scenario III** Costs — high Tangible Benefits — low Customer Base — large Number of Items — many	**Scenario IV** Costs — low Tangible Benefits — high Customer Base — large Number of Items — many

are required, which may include up to two application servers and four database (i.e., storage) servers for the monitoring and portal components. Server hardware costs may be based on current market values given relatively stable price dynamics.

For the context-aware insurance service, a professional appraiser would determine the value of each high-value item. Including initial training, the appraiser could tag the items and install the base station. The installation cost per customer depends on the number of items per customer, the hourly wage of specialists for the appraiser (based on the level of the country where the items are located, as installations have to be made on site), and an estimated time frame between five and twelve hours for installation of the system at the customers site. Furthermore, the recovery rates of stolen items have to be estimated.

Additional variable costs may include maintenance for further software development, system support, employee training, and mobile data communication charges. Mobile data communication charges depend on the amount of information being transmitted. The base station would require more information being sent at less frequent intervals while the GPS tracking module would send less information more frequently. Based on calculations from existing systems a GPS tracking node transmitting its location every ten seconds, 24 hours a day, seven days a week would amount to a total data volume of less than ten megabytes per month per item [12].

Combining the above estimates in the four different scenarios with the internal variables of an insurance company allows for the quantitative evaluation of a specific real-world business case.

5. Conclusions. So far, the potential of context-aware computing has been recognised only in a few industries such as manufacturing or retail in which some context-aware applications have been introduced. In several other industries, however, context-aware computing has experienced just very limited exposure yet. This paper discusses how the innovation potential of context-aware computing can be introduced to these industries by following a rather business-oriented instead of a technology-driven approach. To this end, we introduce a novel procedural framework that is key player-centred and allows for identifying the value of context-aware services. Based on a case study from the insurance industry, we have demonstrated the applicability of the framework with respect to a context-aware service for wireless monitoring high-value items, such as paintings, sculptures, or antiques. This service includes the surveillance of high-vale items, including tracking their location and environment conditions. As required in a multi-sided market, the realisation of this context-aware service is beneficial for each involved key player, i.e., the benefits are shown not only for the owners of high-value items, but also for insurance companies and further stakeholders.

This work thus contributes to the fields of context-aware computing and services innovation on emerging computing and communication systems in three ways: Firstly, the paper proposes a key player-centred framework that is predominantly business-value-driven and, thus, is more suitable for industries such as the insurance industry. Secondly, we discuss a context-aware service that involves several key players, and that provides a

viable solution to the immediate threats confronting insurance companies. Their customers would benefit from an increased protection of their high-value items. Furthermore, a potential reduction in premium prices could be realised due to a network effect of overall reduced risk. From the viewpoint of the customer, the relationship between insurer and customer may shift from traditional paymaster to a problem-solving, trust-worthy helper. Both, insurer and customer, would benefit from an improved quality of service. Thirdly, from a methodological point of view, the solution has potential of being readily adopted by market players. Experience has shown that the full potential of a context-aware computing solution can be effectively determined through the development of a pilot project. Pilot projects support the understanding how context-aware computing solutions may effect revenue and cost of business, and allow for detailed, quantitative business case evaluation. Furthermore, pilot projects may assist in determining how potential users perceive the benefits of the system, and whether or not they have reservations due to possible privacy implications or psychological constraints (e.g., for having high-value items tagged).

Future work will focus on the development of a real-world business case. Researchers are encouraged to apply the suggested framework in various industries to identify new, valuable business cases. Concluding, it will be interesting to see to what extent companies will ultimately embrace the technology and use it as a means for their differentiation from competitors on the market.

REFERENCES

[1] Art Loss Registry, *Website of art loss registry*, http://www.artloss.com/. Accessed 12 August 2013.

[2] L. Auer, E. Belov, N. Kryvinska, and C. Strauss, *Exploratory case study research on SOA investment decision processes in Austria*, in Advanced Information Systems Engineering, Lecture Notes in Computer Science, H. Mouratidis and C. Rolland, eds., Volume 6741, 2001, pp. 329–336.

[3] C. Bauer, P. Dohmen, and C. Strauss, *Interactive digital signage: an innovative service and its future strategies*, International Conference on Emerging Intelligent Data and Web Technologies (EIDWT 2011) & International Workshop on Frontiers in Service Transformations and Innovations (FSTI 2011), 7-9 September 2011, Tirana, Albania, 2011, pp. 137–142.

[4] C. Bauer and S. Spiekermann, *Conceptualizing context for pervasive advertising*, in Pervasive Advertising, J. Mueller, F. Alt, and D. Michelis, eds., Springer, London, United Kingdom, 2011, pp. 159–183.

[5] V.-T. Do, M. Halatchev, and D. Neumann, *A context based approach to support virtual enterprises*, Hawaii International Conference on System Sciences (HICSS '00), 4–7 January 2000, Maui, HI, 2000, pp. 1–10.

[6] A. S. Dobson, *Leveraging the subtleties of location*, Smart Objects and Ambient Intelligence (sOc-EUSAI 2005), 12–14 October 2005, Grenoble, France, 2005, pp. 189–193.

[7] D. W. Engels and S. E. Sarma, *Standardization requirements within the RFID class structure framework*, MIT Auto-ID Labs Technical Report, 2005.

[8] E. Fleisch and C. Tellkamp, *The challenge of identifying value-creating ubiquitous computing applications*, International Conference on Ubiquitous Computing (UbiComp 2003), Workshop on Ubiquitous Commerce, 12–15 October 2003, Seattle, WA, 2003.

[9] M. Friedewald and O. Raabe, *Ubiquitous computing: an overview of technology impacts*, Telematics and Informatics, 28, 2 (2011), pp. 55–65.

[10] B. Friedman, *Value-sensitive design*, Interactions 3, 6 (1996), pp. 16–23.

[11] P. I. Furseth, R. Cuthbertson, and J. Reynolds, *The value dynamics of service innovation*, ISPIM Conference, 6–9 June 2010, Bilbao, Spain, 2010.

[12] GPSauge, *Website of GPSauge*, http://www.gpsauge.de/. Accessed 12 August 2013.

[13] M. Haas, *The formation process of SME networks: a comparative case analysis of social processes in Austria, Belgium and Turkey*, Deutscher Universitätsverlag, Wiesbaden, Germany, 2007.

[14] A. R. Hevner, S. T. March, J. Park, and S. Ram, *Design science in information systems research*, MIS Quarterly, 28, 1 (March 2004), pp. 75–105.

[15] Interpol, *Works of art: recent thefts*, http://www.interpol.int/Crime-areas/Works-of-art/Recent-thefts. Accessed 12 August 2013.

[16] D. A. Kankainen, *UCPCD: user-centered product concept design*, Conference on Designing for User Experiences (DUX 2003), 5–7 June 2003, San Francisco, CA, 2003, pp. 1–13.

[17] G. Kortuem, F. Kawsar, D. Fitton, and V. Sundramoorthy, *Smart objects as building blocks for the Internet of Things*, IEEE Internet Computing, 14, 1 (January/February 2010), pp. 30–37.

[18] R. Kumar, Y. Lifshits, and A. Tomkins, *Evolution of two-sided markets*, ACM International Conference on Web Search and DataMiniing (WSDM'10), 4–6 February 2010, New York, NY, 2010, pp. 311–319.

[19] A. Newitz, *The RFID hacking underground*, iWired News, Wired 14.5 (May 2006), 2006, http://www.wired.com/wired/archive/14.05/rfid.html. Accessed 12 August 2013.

[20] D. A. Norman and S. W. Draper, *User-centered system design: new perspectives on human-computer interaction*, Lawrence Earlbaum Associates, Hillsdale, NJ, 1986.

[21] A. Oulasvirta, *Finding meaningful uses for context-aware technologies: the humanistic research strategy*, SIGCHI Conference on Human Factors in Computing Systems (CHI 2004), 24–29 April 2004, Vienna, Austria, 2004, pp. 247–254.
[22] Z. Pang, Q. Chen, W. Han, and L. Zheng, *Value-centric design of the internet-of-things solution for food supply chain: value creation, sensor portfolio and information fusion*, Information Systems Frontiers, August 2012, pp. 1–31.
[23] J. Pries-Heje, R. Baskerville, and J. Venable, *Strategies for design science research evaluation*, European Conference on Information Systems (ECIS 2008), 9–11 June 2008, Galway, Ireland, 2008, pp 255–266.
[24] B. Richton, G. Vanucci, and S. Wilkus, *Assisted GPS for wireless phone location-technology and standards*, S. Tekinay, ed., Next generation wireless networks, Norwell, MA, Kluwer Academic Publishers, 2001, pp. 129–155.
[25] D. Schuler, *Social computing*, Communications of the ACM, 37, 1 (1994), pp. 28–29.
[26] S. Shane, *Reflections on the 2010 AMR decade award: delivering on the promise of entrepreneurship as a field of research*, Academy of Management Review, 37, 1 (2012), pp. 10–20.
[27] C. Strauss, C. Stummer, C. Bauer, and A. Trieb, *A networked ubiquitous computing environment for damage prevention: a decision support framework for the insurance sector*, International Conference on Parallel Processing (ICPP 2009) & International Workshop on Design, Optimization and Management of Heterogeneous Networked Systems (DOM-HetNetS'09), 22–25 September 2009, Vienna, Austria, 2009, pp. 276–281.
[28] B. W. Tuckman and M. A. C. Jensen, *Stages of small-group development revisited*, Group and Organization Studies, 2, 4 (December 1977), pp. 419–427.
[29] J. Venable, J. Pries–Heje, and R. Baskerville, *A comprehensive framework for evaluation in design science research*, International Conference on Design Science Research in Information Systems and Technology (DESRIST 2012), 14–15 May 2012, Las Vegas, NV, 2012, pp. 423–438.

Edited by: Ethiopia Nigussie, Andreas Riener and Liang Guang
Received: Sep 1, 2013
Accepted: Sep 28, 2013

DOI 10.12694/scpe.v14i3.857
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 14, Number 3, pp. 155–167. http://www.scpe.org

ELL-I: AN INEXPENSIVE PLATFORM FOR FIXED THINGS

PEKKA NIKANDER[*], VADDINA KAMESWAR RAO[†], PETRI LIUHA[‡], AND HANNU TENHUNEN[§]

Abstract. The Internet of Things (IoT) vision is enticing; each and every "thing" in the world is expected to be eventually connected to the Internet, thereby becoming a part of the "context" within which the applications live. In most of the IoT research, the focus has been in enabling movable things to communicate, including phones, tablets, RFID tags, watches, and jewellery, to name but a few. In such an approach, the things are expected to have their own batteries or receive temporary power over short distance electro-magnetic field. This approach has also dominated the more fixed side of the IoT research, including a large fraction on the work on stationary sensors and actuators, focusing also there on battery-based operations and wireless communication.

In this paper, we introduce an alternative view to the world of stationary Internet-connected things. We argue that a large majority of the fixed or stationary things would benefit from being permanently powered using wireline connections, and while doing so, it becomes natural to use the same wires also for their communication and contextual needs. Such an approach allows the appliances to become part of the the wider application context. With this in mind, we introduce the ELL-i platform, a new open source initiative for provide a low-cost flexible prototyping and production platform for extensible, Power-over-Ethernet based smart appliances. We describe the first ELL-i prototyping board, a number of application concepts, and discuss its business model.

Key words: Internet of Things, Context Awareness, Direct Current (DC) power transmission, Power over Ethernet

1. Introduction.

Making real-life appliances intelligent and Internet-connected not only enriches the amount of environmental information available to applications but also creates the promise of making the environment itself a situation-aware part of the "applications". As we can easily see from the multitude of sensors that are available in today's smartphones, once we realise the Internet of Things (IoT) vision [5] of making each "light bulb" Internet connected [12], the amount of available sensor data is likely to multiply. Furthermore, through making the appliances not only sensitive but also active, the appliances may themselves become more aware of the nearby humans, reacting and pro-acting in a situation-aware manner [29].

Considering physical space, a key aspect of making a room, building, or any physical space context aware is to have a sufficient number of sensors and providing a framework for pre-processing the data [6]. This is typically expected to take place through installing a *sensor network* to an already existing building or space, and routing the collected data to an aggregation service. In the large majority of the recent works, the sensor network is expected to be a wireless one, to the extent that the most widely referred surveys carry the word "wireless" in their title [1, 2].

At the same time, most of the works have implicitly ignored how the sensors are to be powered or assume that they have a battery that will last for their lifetime. Additionally, in most of the published research, the presented use cases and applications assume the use of some wireless connectivity technology, although the primary ideas do not specifically refer to either wireless or wired networking, or to a any specific information and communications technology. Looking at the situation from greenfield applications point of view, or more widely from a sustainability perspective, assuming that the sensors will be replaced every few years, or even every tens of years, does not look like a very rewarding alternative [45].

While it may well be possible to create sensor nodes that scavenge energy from the environment and use radio or light bursts to communicate with the environment [52], creating wireless powered every day appliances that have some other active function than mere sensing seems improbable. Hence, it seems prudent to assume that a large majority of different appliances will remain wired, including air conditioners, blinds, coffee machines, doorbells, electroliers, and freezers, to mention but a few.

In this paper we argue that it makes economic sense to introduce a public open source platform for wired, communicating, intelligent "things". Once established, such a platform would allow designing and producing inexpensive appliances that are able to monitor their environment, participate in wider situation data collection and perusal, and act in accordance with the users' explicit and sometimes even implicit intentions. In particular,

[*]ELL-i open source co-operative, Helsinki, Finland. (pekka.nikander@ell-i.org). Questions, comments, or corrections to this document may be directed to that email address.
[†]Department of Information Technology, University of Turku, Finland. (vadrao@utu.fi).
[‡]EIT ICT Labs, Helsinki Node, Finland. (petri.liuha@eitictlabs.eu).
[§]Kungliga Tekniska Högskolan, Stockholm, Sweden. (hannu@kth.se).

we introduce the ELL-i platform, an open source initiative that aims to provide a low-cost prototyping and production platform for embedding wired intelligence into buildings, appliances, and other implements.

As its first technical platform, the ELL-i initiative provides open source hardware and software for building inexpensive embedded intelligence into devices, allowing them to communicate and be powered with Power-over-Ethernet (PoE) [25], an established but rapidly evolving standard for providing up to 100 Watts of electric power [21] through standard Ethernet cables. The currently available ELL-i prototyping board is compatible with the popular Arduino [7] prototyping platform.

The ELL-i initiative is organised as a co-operative, providing an egalitarian ownership model that anchors the property rights with the community. The governance model is designed to be meritocratic, assuring that the platform will evolve in a technically sound manner. The incentive models are designed to inspirit participation from a wider open source developer community while still allowing mass-market production by commercial companies.

The rest of this paper is organised as follows. First, in Section 2 we consider related work from a relatively wide perspective. Then, in Section 3 we introduce the ELL-i platform from a technical point of view. In Section 4 we briefly describe some example applications that are immediately viable with ELL-i, followed in Section 5 with a succinct description of the business model. Section 6 enlists our ongoing work. We conclude the paper in Section 7, attempting to summarise the lessons learned so far.

2. Related work. Proliferation of cheap and versatile sensors, actuators, computing, storage and networking solutions, augmented with the desire to connect people with their devices in a meaningful way, have contributed significantly to the IoT research. In this work we focus on an almost banal way for providing both the communication and powering solutions to fixed and immovable devices, which would anyhow be powered by wirelines, using Power over Ethernet (PoE) standard.

In this section, we give a general and broadly accepted definition for Internet-of-things (Sect.2.1), explore IoT applications (Sect.2.2), and delve quickly into the power management and security related aspects in IoT and M2M research (Sect.2.3). We conclude by enumerating some other open source embedded PoE platforms (Sect.2.4).

2.1. Internet of Things and Machine-to-Machine. Perhaps the first definition of what later became Internet of Things was given by the Internet pioneer Vint Cerf in this ACM conference talk in 1997 [12], where he envisioned each light bulb having their own Internet address. The term Internet-of-Things (IoT) was first used by Kevin Ashton in 1999, when he referred to identifiers placed in objects so that they could be monitored and managed by computers [4]. It became related to the famous and more generic definition of the disappearing computer by Mark Wiser [53], where he described that miniaturization and ubiquity of sensors would eventually lead to disappearing to the computational elements "into the fabric of everyday life". This has been later denoted to as pervasive or ubiquitous computing.

More recently, one of many definitions of IoT is given by the Cluster of European Research Projects (now IERC), who defined IoT as a "dynamic networked global infrastructure with self configuring capabilities based on standard and interoperable communication protocols where physical and virtual things have identities, physical attributes and virtual personalities and use intelligent interfaces, and are seamlessly integrated to the information network" [51].

A second dimension and branch of development of IoT is the so called Machine-to-Machine or M2M concept, which further refers to linking the plethora of digital devices (from simple microcontrolled devices to smartphones and large industrial instrumentation). This refers more to the communications protocols required to allow devices to communicate with standard communications, typically wireless or even more specifically cellular systems. A couple of popular projects developing the technology are the Eclipse M2M Industry Working Group [18] and ITU-T Focus Group M2M [27].

2.2. Internet of Things applications. IoT applications include a variety of applications ranging from tracking and tracing of objects to applications applying wireless sensor networks and Machine-to-Machine (M2M) systems. Typical applications are in military use cases (like intelligence, surveillance), environmental use cases (e.g. monitoring conditions that affect farming and animal husbandry, or monitoring macro-scale phenomena in soil, forest, marine or atmospheric contexts), health applications (remote monitoring of patients,

tracing of care personnel or equipment), home applications, or other commercial applications, such as interactive smart spaces in office or other working environments.

The use cases include applications with large area networks and high mobility but also no-mobility use cases. Wu et. al. discuss the requirements for future applications from the M2M perspective [54]. While many of them assume wireless connectivity, a good part of them are in fact for fixed things, like in home management, industrial automation and metering, sensors, and lighting. More generally, there are IoT use cases with no mobility in smart homes or smart buildings and smart cities; e.g., for the purpose of reducing the consumption of resources associated with buildings (electricity, water) or improving the satisfaction level of humans populating them [39].

2.3. Power management and security in IoT and M2M research. In this work, IoT devices have been classified by the amount of power required for their efficient operation. Some IoT devices use batteries, some scavenge energy from radio waves (RFID and NFC), while others are powered by separate power supply from the mains. Smart and tiny embedded systems have sensors, actuators, CPU, memory and a low-power communication device [34]. Usually, these can be powered by a battery. On the other hand, Wireless Sensor Networks (WSN) use energy efficient battery powered wireless devices for transmitting sensor data [2].

From the security point of view, as the complexity of the encryption algorithm increases, their battery consumption increases as well. It has been shown that among symmetric encryption algorithms used in security protocols, the AES128 algorithm increases the battery consumption and the processing time by 75% and 65% respectively when compared to "no encryption" results [20]. Furthermore, increasing the key size for the encryption also increases the battery consumption and processing time.

2.4. Open source embedded platforms. Today, there are a few tens of slightly different open source embedded platforms. The Raspberry Pi [42] forms a category of its own, offering an unbeatable price point in the embedded Linux category. In the lower category, the Arduino [7, 23] project has gained the largest user community, with half a dozen different official boards and a number of derivatives.

The Arduino project is a somewhat loose collaboration led by a number of people that developed the original Arduino design. They have decided to open source all of the software and hardware but to keep the Arduino and associated graphics as a trademark. At the moment there are three companies that are authorized by the project leaders to produce official Arduino boards [37]. As another example, the Raspberry Pi is produced by the Raspberry Pi Foundation, which is a UK registered charity supported by the University of Cambridge Computer Laboratory and Broadcom [28]. For its first three years of operation, the foundation income was annually less than £10.000. The foundation income and spendings for 2012 were not registered at the UK Charity Commission at this writing.

There are other Arduino and Raspberry Pi based PoE solutions [11, 47, 48, 49] currently available in the market. Some of them are complete PoE solutions, while others are shields that can be plugged into an Arduino. The Arduino Ethernet with PoE [48] is an Arduino development board with the built in Wiznet Ethernet interface. The Arduino Ethernet Shield with the PoE Module [47] allows an Arduino board to connect to the Internet. It is based on Wiznet W5100 Ethernet chip which provides a network IP stack capable of both TCP and UDP. The PoEthernet Shield [49] gives access to the Internet via the Ethernet library and also allows the Arduino based projects to power itself from the Ethernet line. The Raspberry Pi Power Over Ethernet (PoE) Adapter [11] is designed to be plugged directly on to a Raspberry Pi and power it with +5 Volts at up to 1.0 Amps output.

All the above solutions focus on providing power to the MCU and its digital peripherals, powered through the 3.3V and/or 5V systems. The long term vision for ELL-i is significantly different. ELL-i is a multi-voltage domain design making the "raw" 48V from PoE available. ELL-i is aiming to be an inexpensive platform for low-cost PoE devices that come with integrated cloud services.

3. ELL-i platform architecture. In this section, we describe the ELL-i technical architecture in sufficient detail to ground the discussion that follows[1]. First, we describe the overall technical architecture in a bottom-up-fashion, including hardware, firmware, and the forthcoming cloud components. After that, we introduce the current implementation. We conclude the section with a brief discussion of the design choices made so far.

[1]For further details, please refer to the open source design files at `http://www.github.com/ELL-i`.

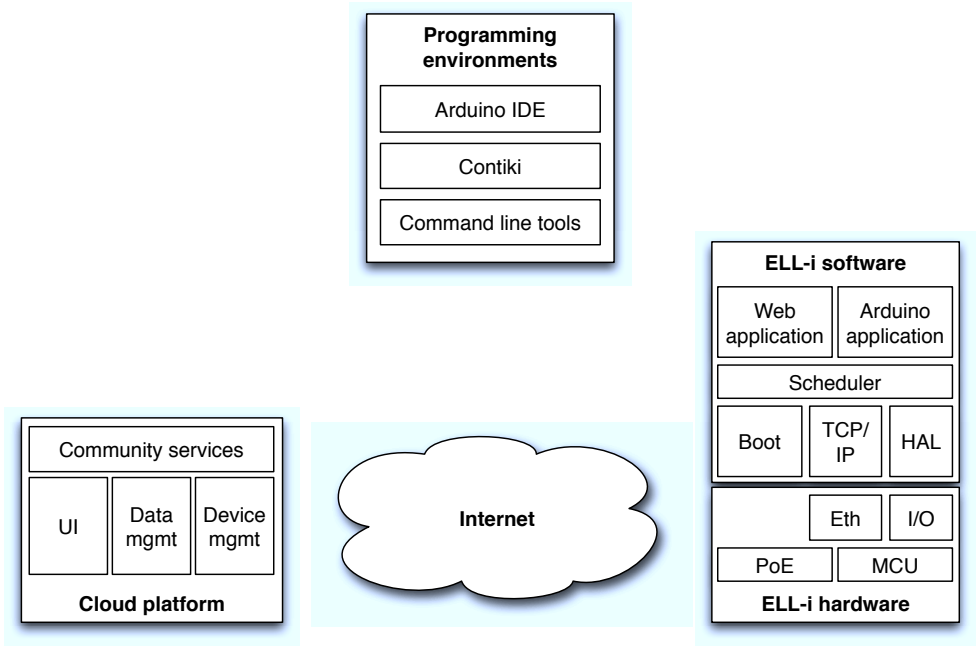

FIG. 3.1. *ELL-i platform architecture*

3.1. Overall technical architecture. The ELL-i platform consists of a hardware architecture, a runtime (firmware), cloud-based components, and support for a set of programming environments. The basic architecture is depicted in Figure 3.1.

The hardware architecture is based on combining power and communications on a single cable, making both of them available to applications. In practise, the current implementation is based on the IEEE 802.3at [25] Power-over-Ethernet standard, which is an economical solution when the power consumption is roughly in the 1–100 Watts range. In the long run, we expect to expand the power range to both smaller and higher power levels; for example, there are no technical reasons why Power-over-Ethernet technology could not be applied to thicker wires and higher currents.

The main goal of the runtime and the cloud components is to make it easy to provide connectivity for the appliances. Hence, the goal of the runtime is to package TCP/IP based protocols, such as HTTP [19] and CoAP [46] in an easy-to-use way. As it is undesirable to perform packet processing in an interrupt context and as many of the protocols depend on timeouts, abstracting the protocol processing away from the runtime API requires some kind of threading.

ELL-i is positioned as an easy-to-start platform. Hence, we have focused on providing an Arduino IDE extension that allows Arduino skills to be readily applied on ELL-i. However, we also realise that the default Arduino runtime is geared for hobbyists. Therefore we will also support XCode and Eclipse.

3.1.1. Hardware. The platform hardware consists of an MCU, an Ethernet communications module, a power supply module, and an I/O subsystem that connects the system to the real world. The MCU provides the smarts, including a computational core, non-volatile (flash) and volatile (RAM) memory, and peripherals for connecting with the communications, power, and I/O subsystems. It executes a program stored into its flash, converting commands received over the Internet into changes into its I/O state and vice versa. The MCU program is primarily responsible for implementing the TCP/IP protocols. In a typical case, it takes also (partial) responsibility for controlling the DC/DC power supplies, thereby reducing the overall cost of the system.

The Ethernet communications module implements the IEEE 802.3 [24] communications standard, sharing the same cable with the PoE-based powering of the device. It consists of three components: the isolation

magnetics, the physical interface (PHY) and the media access control interface (MAC). The firmware takes care of the actual packet processing.

The power supply module consists of a Power-over-Ethernet signalling module and a DC/DC power supply module. The former implements the IEEE 802.3af or 802.3at Power-over-Ethernet standard, including the physical-layer signalling that consists of an initial resistive-load based detection phase, a current-sink based classification phase, an optional second classification mode added in 802.3at, and an operating-voltage ramp-up phase.

The DC/DC power supply is responsible for converting the incoming Power-over-Ethernet 48V voltage into the 5V and/or 3.3V needed by the MCU and other digital electronics. In a specific ELL-i application, there may be an additional power supply producing "bulk" power for the "real world" components, such as a constant-current PSU for a high-power LED light, a circuit generating suitable current pulses for a solenoid, or a precisely controlled PSU generating modulated currents for a direct-current driven electric motor. In most real world applications there is a difference between the power requirements of the digital electronics, which typically require constant voltage but variable current, and that of the real-world actuators, which typically require controlled current but are relatively insensitive to the actual voltage.

Finally, the I/O subsystem adjust the electrical signals generated by the sensors and required by the actuators with the levels created and tolerated by the MCU. In a typical case, the I/O subsystem consists of operational amplifiers acting in amplifier and/or level shifter role. In many cases also opto-isolators or other isolators are needed.

3.1.2. Runtime. The runtime system forms the basis of any firmware running on the ELL-i platform. It consists of a software library that is linked into any specific firmware build by an application developer. The runtime provides a number of APIs and services that make application development easier than building everything from scratch.

In ELL-i, the main goals of the runtime are to provide an easy-to-start but still powerful programming environment for both novices and professionals, and to offer facilities for communicating with the cloud components in the Internet. To achieve that, the ELL-i runtime consists of a boot module, a hardware abstraction layer (HAL), a tiny scheduler providing concurrency, an Internet-communications module, an optional built-in Webserver, and a set of APIs providing programmatic access to all of these.

The set of APIs may be divided into five groups, corresponding to the booting, HAL, scheduler, Internet, and Web-services. In the current implementation, the boot and HAL APIs provide an Arduino-compatible programming environment, allowing programmers to initialise the hardware using the Arduino `setup()` function and to run a main loop within the `loop()` function. Both of these are called by a `main()` function, allowing more advanced programmers to override the Arduino approach. The familiar Arduino HAL APIs are supported, such as accessing GPIOs and serial lines.

While the original Arduino runtime does not provide any concurrency, thereby causing e.g. busy loops to stall the whole MCU, in the ELL-i runtime there is a small scheduler running "under" the Arduino APIs. Using the scheduler, the Internet and Web-server modules are implemented so that they always run concurrently with any Arduino sketch.

The scheduler API provides services for starting and stopping concurrent threads and adjusting the scheduling algorithms. By default threading is fully pre-emptive, allowing the TCP/IP stack to work on a higher priority than any Arduino sketch. Any I/O modules requiring precise timing need to run on their own threads, or in their interrupt context, in order to preempt packet processing. Dynamic memory management is not supported and is strongly discouraged; any threads need to have their stacks to be allocated at compile time.

The Internet API provides services for operating TCP, UDP and CoAP [46]. The TCP/IP stack is based on uIP [15]. We support Contiki [16] style protothreads [17], thereby making it easier to support existing uIP-based protocol stacks and applications. The optional built-in Web-server allows the ELL-i platform to provide AJAX-style Webservices towards the local network. For security reasons, the TCP packet lifetime is limited to one hop, thereby making the webservice unreachable from the Internet even in the case the node has a public IP address and full Internet connectivity.

The runtime library has been carefully built in a way that allows the linker to leave out any modules that are not used by a particular application. However, whenever the default `main()` function is used, the runtime

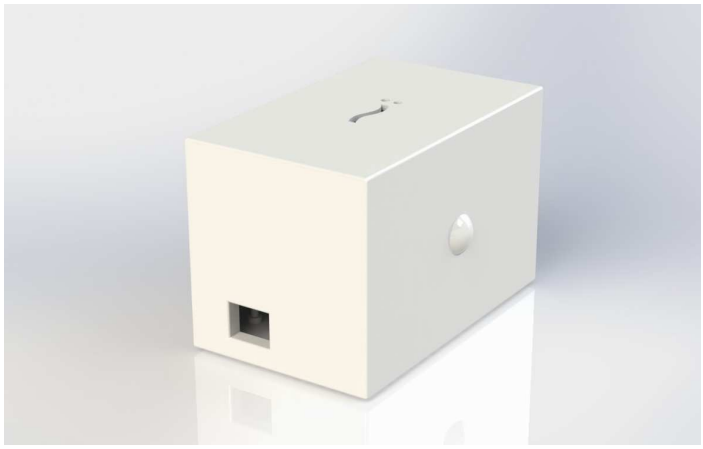

FIG. 3.2. *ELL-i case design*

includes the UDP/IP stack, the CoAP protocol, and a basic service module that registers the node with the cloud components and by default sends sensor data to the cloud.

3.1.3. Cloud components. In addition to the software components running on the ELL-i hardware, the ELL-i platform contains also cloud components, running on a compatible cloud runtime, such as Amazon Web Services (AWS). The cloud components provide interaction, update and community services to the ELL-i boards, and they are enabled by default in each ELL-i development board.

In the default configuration, a newly installed ELL-i development board attempts to contact its counterpart component at the cloud side, making the cloud-side component aware that the board is running and has Internet connectivity. At the same time, the user may direct their browser to a board-specific URL and view the board status. The board may also periodically send sensor information to the cloud, such as a reading from the MCU-internal temperature sensor. This data is then visualised to the user.

Another cloud-based service is the ability to remotely upgrade the firmware on ELL-i boards. If the remote upgrade is enabled, the firmware contains a tiny CoAP-based communications module that is able to download new firmware page-by-page to the RAM, and flash it. By default, each page must be protected with a symmetric cryptographic key, shared between the board-specific cloud component and the board itself. If the remote upgrade fails and the board no longer boots, it still remains possible to completely re-flash the board using a serial line and a separate programming board.

3.1.4. Programming environments. The ELL-i platform is designed to support multiple programming environments, including the Arduino, Eclipse, XCode, and for the old timers, plain `emacs` and `make`. At this writing only the Arduino IDE is formally supported, even though some of the early adopters do use command-line tools. The aim is to make it *easy* to move from the Arduino IDE to the more advanced programming environments, with clear instructions on how to convert Arduino sketches into proper C++ source files. This also requires clear explanations of what is happening under the hood in the runtime.

On the compiler side, at the moment we are still using the very GCC version provided by the default Arduino IDE. However, our plan is to move to LLVM [32] as soon as possible, and to enable global link-time optimisation. This together with well-written static initialisation of global C++ objects, such as the Arduino `Serial`, allows the compiler in many cases to optimise away not only virtual function calls but use constant propagation down to the level where the Arduino syntactic sugar may be optimised completely way, providing the same level of efficiency as would normally be achievable through bare-metal access while still working with high-level C++ abstractions. We are also considering whether it would be possible to pre-run the Arduino `setup()` function and some of the preceding setup machinery during the compilation time [43], providing boot-time optimisations.

3.2. Current implementation. At this writing in September 2013, the currently available implementation consists of the first prototyping board, an initial runtime, and support for the Arduino and Contiki APIs.

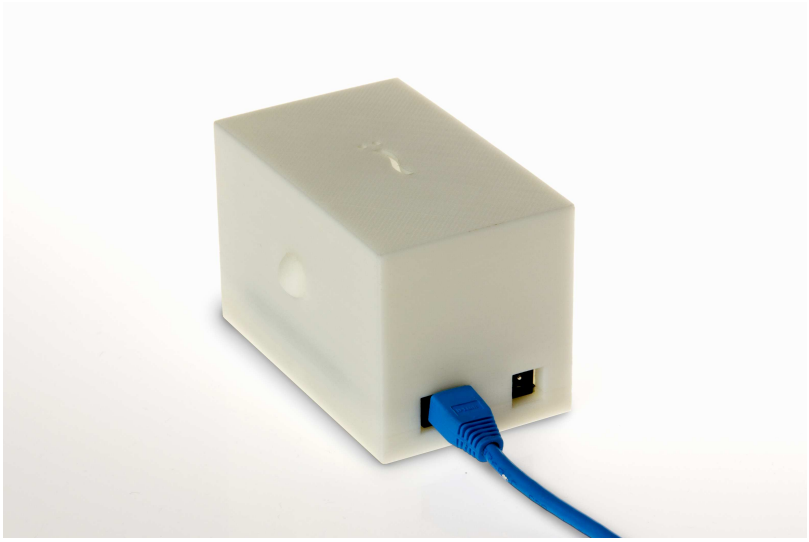

FIG. 3.3. *ELL-i 3D-printed prototype case*

Work on the cloud-side components is to be started. The goal of the first prototyping board was to create Arduino-compatible hardware and software that would support a more powerful MCU than the present basic Arduino's do and to allow custom DC/DC power supply design. This goal has been reached. With the first prototyping board, prospecting developers can utilise the existing Arduino examples, learning material, and shields, to a large extent.

The first ELL-i prototyping board was designed during spring 2013. These initial boards have an STM32F051 [50] ARM Cortex-M0 microcontroller, Microchip ENC28J60 [38] Ethernet chip, Linear LTC4267 PoE controller with build-in flyback SMPS controller, and Arduino-compatible headers. We also have a 3D printed case for the prototyping board, with enough of space for a few Arduino shields. The current computer generated case design is depicted in Figure 3.2 and a corresponding 3D-printed prototype in Figure 3.3.

For Arduino-compatibility, we designed the LTC4267-based flyback DC/DC converter to produce 5V, using a standard linear regulator to produce 3.3V out of the regulated 5V supply. That allows us to replicate Arduino Due and Arduino Mega I/O pins with some accuracy. The I/O pins have been assigned so that there is a separate timer/counter channel available for all of the digital I/O pins, an ADC channel for all of the analog pins, and the DAC on the STM32F051 is connected to the Due DAC0 line.

As the first programming environment, we have been focusing on supporting the Arduino IDE. Starting with the Arduino IDE 1.5 extension mechanism [35], we have developed our own Arduino-compatible runtime that compiles about over 90% of the sketches provided with the Arduino IDE. However, while most of the APIs are supported at the syntax level, there is still quite some underlying code missing.The existing functionality is enough for most of the basic sketches, though, thereby helping people to get started with ELL-i.

3.3. Design choices. With the ELL-i platform, our goal has been to produce an easy-to-use, flexible, and inexpensive platform for fixed appliances. These goals have guided our design choices. Indeed, the primary choice of using Power-over-Ethernet stems from this goal, as it allows us to use a single cable for both power and communications, thereby potentially reducing the overall cost while still being flexible and easy-to-use.

Considering the component choices in the first prototyping board, we have tried to select the lowest-cost components that are easy-to-use and flexible enough. Consequently, we opted for the lowest end ARM series, Cortex-M0, as they are easy-to-use as a familiar 32-bit platform and most likely to hit the lowest price point in the area within a few years[2]. The chosen Ethernet chip is the lowest-cost denominator of those widely used

[2]At the moment, there are still older Cortex-M3 based MCUs on the market that are sold cheaper than the newer Cortex-M0 MCUs in small lots. However, the lowest price point is most likely to be replaced with the Cortex-M0 MCUs within the next couple

in the market. At the PoE and DC/DC PSU side there is still room for optimisation, but we are currently planning to leave that for later.

For hardware-level expandability, we are still working to determine the right approach. At the moment, we are using the Arduino-compatible I/O connector design for development-time extensions. However, it is clear that they are not a good choice for production use, for many reasons[3].

Considering the connectors and mechanics, our goals are again the same: easy-to-use, flexible, and inexpensive. For Ethernet, the goal is to support both the familiar RJ45 connectors and the punch-down connectors for fixed design, the punch-down connectors being clearly cheaper. In the new design, we are looking at various options for the connectors between the production and development boards. At the moment, we are planning to allow different approaches for tying the boards together, including using screws and having suitable notches in the packages.

At the firmware side, it looks likely that we will end up implementing our own runtime. Looking at the various open source RTOS options available, they typically include more than what we need and with a license that is not fully compatible with our planned business model; see below. The cloud-side components will be based on the lowest common platform-as-a-service (PAAS) denominator, which at the moment is AWS APIs, also supported by e.g. the AppScale [13] open source framework.

4. Application examples. As a platform, the ELL-i design is open ended. In this section, we briefly describe a few designs that we presume the ELL-i platform making easier or less costly to implement than before. Furthermore, wherever applicable, we attempt to illustrate how the open source nature of ELL-i will benefit the individual applications.

4.1. Illumination. The very first demo that we implemented using ELL-i was a driver for a commercial high-power RGB LED projector. Since then, we have repeated the LED driver exercise a few times.

From the electronics point of view, an adjustable high-power LED driver is a relatively simple device, as LEDs in general are current-driven components. That is, the voltage drop over a LED remains relatively constant independent of how much current is flowing through the LED, and the amount of current determines how brightly the LED shines. The typical voltage drop over a high-power LED is around 3V. Hence, for driving a LED from a voltage source that is higher than the voltage drop of the LED it is enough to limit the amount of current. For indicator LEDs the current is usually limited with a suitable series resistor; however, such an arrangement would be very inefficient for driving a high-power LED out of the PoE nominal 48V. Fortunately, a buck converter, the simplest of the so-called switching mode power supply (SMPS) [36] topologies, is quite easy to regulate to produce a constant, adjustable current. All that is needed is a simple current-measurement circuit and a simple algorithm to drive the switching FET in within the driver. While achieving high efficiency requires somewhat more work, even an electronics amateur, like us, is able to design a simple MCU-controlled buck converter in a couple of weeks time.

So, from the hardware point of view, driving LED-based lamps would be an ideal application for the ELL-i platform. In addition to the components already in ELL-i, a LED driver at its simplest form requires only few inexpensive components, including a switching FET, a choke coil, and a diode. Consequently, ELL-i may be an excellent platform for implementing future, more intelligent show lighting systems, flexible retail lighting, and other illumination applications where it is beneficial to control each and every lamp separately. In such an environment, ELL-i implements Vint Cerf's vision of each light bulb having its own IP address [12].

To communicate with the external world, and ELL-i node may implement the Ethernet or TCP/IP variants of the commonly used lighting control protocols, including DALI [14] and DMX [9].

4.2. Automation. Another area where the ELL-i platform may bring benefits is building and industrial automation. In the building automation area, there are already a number of Power-over-Ethernet products on the market, including locks and other security and safety equipment. However, to a large extent the existing

of years.

[3]The Arduino-compatible connectors are physically quite large, requiring more space than what is available within many devices and e.g. within a pattress box. Furthermore, the Arduino-compatible connectors are basically unbuffered, making it somewhat tricky to get good ADC readings or driving higher loads through them.

automation systems are still based on standards, such as BACnet [10], KNX [30], LonTalk [26], or ModBus [40], which generally utilise twisted pair control cables and separate power supply.

From that point of view, ELL-i provides an open source platform to build upon. As such, ELL-i provides little benefit directly to automation, as it is lacking direct connectivity with sensors and actuators, and it does not as such implement any of the existing protocols. Therefore, as a standalone component it does not integrate with the existing automation systems.

Being a small, connected, intelligent, power-convoying low-cost component, ELL-i could potentially be co-located with the sensors and actuators, providing power to them and integrating both intelligence and connectivity directly to them. Consequently, it would allow replacing the hierarchy of separate field devices (sensors and actuators) and the programmable logic controllers (PLC) with a flat logical hierarchy, allowing the physical connectivity to be arranged in any topology.

Beyond the basic functionality challenges, applying ELL-i in automation systems would require a number of non-functional challenges to be addressed. First and foremost, it should be demonstrated that the ELL-i system is reliable as the known and proven old solutions. As it is unlikely to happen as such any soon, one way to address the reliability concern is to build reliability through redundancy. That is, a system consisting of the ELL-i platform based nodes can be engineered to be inherently redundant, placing two ELL-i nodes at each sensor or actuator side, using standard Ethernet network redundancy approaches[4], and addressing software-level redundancy through differing runtime systems.

4.3. Education. A third area where ELL-i may turn out to be useful is education. As an open system, ELL-i itself can be studied and modified by the students, and it could as such be extended to be a platform for teaching basic programming. Secondly, as an easy-to-use embedded system ELL-i may be used in various student projects where real-life interaction is needed.

4.3.1. ELL-i platform itself as course material. The ELL-i platform has been designed to be not only easy to use but also easy to understand. The hardware represents a complete design with an MCU, Ethernet-based communications, and a DC/DC PSU. The chosen MCU is a relatively simple but still a powerful one, from a family that is commonly used in open source projects. The CPU core within the MCU is a modern 32-bit RISC architecture with simple 3-stage pipelining and no instruction reordering, being easy to teach and to understand. The MCU-related part of schematic is very similar to the relevant part of the MCU manufacturer's evaluation board. The Ethernet communications module consists of a single chip and the magnetics and has a wide support in the open source community.

At the software side, the firmware has been written in a modular manner, as was explained above in Section 3.1.2. Hence, studying simple firmware designs that do little but initialise the MCU and blink some LEDs is relatively easy. The modularity then allows more complex features, such as communications and scheduling, to be added in a piecewise manner to the design. This allows the teacher to demonstrate the full runtime in a gradual manner, starting from MCU initialisation and simple I/O, continuing through the SPI and I2C protocols up to simple Web services.

Hence, the ELL-i platform may be used to introduce and demonstrate all the elements of a modern Web-connected computer, including both the hardware and software, without the complexities.

Given that the Arduino APIs and sketches are C++ programs in practise, it would be possible to write a set of C++ proxy classes and templates that compile an Arduino sketch so that it runs in an emulator, allowing the functioning of such a sketch to be simulated and illustrated on a PC. Furthermore, with the Ethernet interface, another set of proxy classes could be used to instrument a sketch, allowing it to be run on the ELL-i platform while simultaneously inspecting the variables and I/O ports. The instrumentation classes would communicate the application state over the Ethernet to a PC, which would then use the simulator user interface to illustrate the application state.

4.3.2. ELL-i as a student project platform. Due to the simplicity and the modular design of the ELL-i platform, it is a relatively easy platform for students to build their platforms upon. Arduino compatibility allows adoption of almost any of the some 300 existing Arduino shields, typically with some minor software-side modifications. In many cases, the more powerful MCU on the ELL-i platform allows the sensors or other

[4]Dual cabling with redundant switches, spanning tree protocols to avoid bridging loops, etc.

components on the shield to be used more intelligently than with the basic Arduino boards. Ethernet-based powering allows the devices to be placed relatively freely anywhere, thereby allowing the students to build smart spaces or place smart fixed devices at desired locations.

The simplicity and flexibility of the platform also allows the students to replace many parts of the platform with their own components. For example, at this writing, a student group at University of Turku have completed a preliminary port of the FreeRTOS [8].

EIT Smart Spaces summer school. In June 2013, a couple of weeks after the first ELL-i prototyping boards had arrived, two of us held a one-day ELL-i tutorial at the European Institute of Innovation and Technology (EIT) Smart Spaces summer school in Grenoble, France. The tutorial lasted for six hours, within which the computer science students built a simple buck converter based LED driver, using the ELL-i platform, discrete components, and the Arduino IDE.

During the tutorial, the students learned how to handle basic physical electronic components, including resistors, capacitors, coils, FET transistors, diodes, and LEDs. Starting from a very simple design, they built in a stepwise manner an adjustable constant-current DC/DC converter that generated some 300 mA of current from the nominally 48V unregulated DC power available from the PoE header on the ELL-i prototyping board. The background of the students varied from ones that had only taken theoretical basic courses in electronics, as a part of their basic studies, to a few that had built simple Arduino-based prototypes before. None of the students had previous knowledge of how the DC/DC power supplies actually work.

5. Business model. In this section, we briefly look at the ELL-i business model, which somewhat deviates from most typical business models, being closer to the Linux kernel community [22, 33].

As stated, the goal of the ELL-i project include providing an inexpensive and easy-to-use platform for all kinds of fixed appliances. As always, in electronics production achieving a low price requires high volumes. Hence, the ELL-i business model aims to make the platform widely available at a low cost, thereby encouraging it to be widely used, leading to increasing volumes and thereby even lower prices. In other words, the project does not aim to create profits as such but to distribute the added value in the form of lower prices, thereby contributing to the economy as a whole.

The ELL-i project itself has been incorporated as a co-operative, with the intention that the hardware and software copyrights are co-owned by the developers. That is, the aim is that the co-operative owns the copyright for all of the hardware and software components within the core ELL-i platform and the developers who contribute to the project may join the co-operative as members if they wish to do so.

We envision that the project and the co-op will create interest among commercial for-profit companies. This is important both for creating volumes and for distributing the created value to the economy. Hence, the ownership, governance and licensing models have been planned and will be adjusted to make the platform lucrative for commercial licensees.

In order to get the platform there in the first place, we first need to entice a group of enthusiastic developers that want to contribute to the project. For that, we presume that we first need to sell a few thousand boards. We expect that 1–4% of the early adopters will turn into active developers [3]; see Section 5.1 below.

The ELL-i business model needs to be balanced between the early market need of creating a sizeable developer community and the somewhat later need of attracting commercial companies in order to create volumes and benefiting the larger society.

The resulting structure has probably a relatively low monetary value compared to its overall utility value, as the ownership will be more anchored than in most alternative models. However, we consider this as a virtual from the macroeconomic perspective [41], somewhat similar to what state-owned enterprises provide but detached from the populist political system.

5.1. Early market challenges. At the moment, the main challenge ELL-i faces is a marketing one. The first version of the hardware platform exists both as open source schematics and layout and as actual, functioning hardware boards. The software platform is being continuously enhanced, providing the basic facilities that are needed to get started. However, the group of active developers is small, just a handful of people. In order to get the ball rolling, more active developers are needed.

We believe that the ELL-i approach of providing both power and communications over a single cable is beneficial to a fair number of people, especially when the incoming power can be easily converted into a DC

voltage suitable for various kinds of loads. Hence, from the ELL-i point of view, a major challenge is to find the people that are already struggling with the appliance powering problem and demonstrate the viability of the ELL-i solution. That in turn requires much more visibility than what ELL-i enjoys today and a number of additional power supply designs, making it easier to have suitable power tailored for each load.

An initial market presence may be achieved through crowd-funding [31]. That may help with gaining initial production volume and inducing a few new developers; however, such a campaign is also likely to saturate the early market, making it considerably harder to sell new hardware for a while. Hence, at the time a crowd-funding campaign is commenced, there must be sufficient structure in place so that the new developers have a clearly laid approach and incentive to start contributing to the project.

What comes to the powering of appliances, we basically need to learn ourselves – within the core group – to build a more diverse set of power supplies. We also need to make the resulting knowledge available in a form that is easier for relative newcomers to apply than the currently available information, initially as recipes and later on as ready-applicable DC/DC converter units. The basic ELL-i recipe for providing adjustable constant-current power is already there, thereby enabling adjustable LED drivers, and we expect soon to complete our first design for inductive loads, such as solenoids and motors.

5.2. Business model considerations. As a first approximation from the conventional economics point of view, the ELL-i platform seems to form a fairly classical two-sided market [44]. ELL-i offers the platform essentially free-of-charge to the developer community, allowing individual developers to innovate and fulfil their personal needs. At the same time, it offers the platform for commercial companies for a fee. At the same time it attempts to form a joint community of both individual developers and product companies, creating a larger two-sided market where the sides are the hardware component manufacturers and the product users. From that point of view, ELL-i benefits the component suppliers through larger volumes, the product companies through increased bargaining power, and the product users through increased variety and lower average price of the products.

From a societal point of view, if ELL-i succeeds as a platform it may produce essentially a virtual non-profit monopoly, benefiting the society through the increased efficiency of scale of the used components and overall lower user prices of the technology. At the same time, its egalitarian ownership structure anchors the forming intellectual capital within the community while the meritocratic governance structure allows any further development to be primarily technology driven, aiming for long-term engineering excellence instead of short-term profits.

6. Ongoing work. As a platform and a business model, ELL-i is still at its infancy. As a development platform, we are still busy laying out the building blocks: the next version of the hardware is being worked on, the software is receiving constant improvements, and at the tools side we are smoothing out the developer path from the easy-to-start-with but simplistic Arduino IDE to the more powerful full-fledged tools. At the same time we are looking at how to expand the technology to convert a wider variety of environments; for example, it would be very desirable to use the ELL-i platform without requiring any new cabling when renovating existing installations.

From the community building point of view, ELL-i has but started. We are still considering how to best launch a campaign to expand the developer base beyond the founders of the co-operative. It seems clear that we need some sort of a crowd-funding campaign to attract developers and to gain initial volumes.

Finally, the governance and licensing models clearly need much more work. Our understanding of how to structure the business model in a way that allows both fast organic growth and sustained technical development is still inadequate, requiring both better theoretical understanding of the various stakeholder incentives and real-life experimentation in the form of trying out a few alternatives.

7. Conclusions. Realising the Internet-of-Things (IoT) and Machine-to-Machine (M2M) visions will make our environment immensely contextual in the sense that the digitally interconnected artefacts in our environment may jointly form a rich model of what is taking place in any given space, allowing the environment to react in a smart manner. In this paper, we have discussed one potential step towards that vision, the ELL-i initiative and technical platform.

The ELL-i initiative is organised as a co-operative, with the intention of producing an inexpensive open

source hardware and software platform for wired appliances and implements, such as aquaria, bowling alleys, cooktops, dishwashers, all the way to toasters, underlayments, vacuum cleaners, washers, xerox machines, yachts, and kitchen zinks. The currently available ELL-i prototyping board allows researchers and other developers to explore the ELL-i approach, creating prototypes of intelligent, communicating things that can be readily located anywhere at the distance of an Ethernet cable. With its Arduino compatibility, the ELL-i platform provides a very low entry barrier, while still outlining a clear path for more powerful approaches.

From the technical point of view, the main promise of the ELL-i approach is in its ability to provide reasonable amounts of electrical power, combined with the communication and data processing capability, at a very reasonable price point. If ELL-i ever reaches mass-production quantities, it will provide an electronics module that allows almost any wired electrical device to be converted into its "smart" equivalent at the additional production cost of just a couple of euro. Consequently, as even a typical light switch or toaster is priced at a level close to or exceeding ten euro, that may gradually enable complete "smartification" of our urban environment.

The ownership, governance, and incentive models built into the ELL-i co-operative have been designed with sustainability in mind. They aim to create an openly governed, jointly owned platform that is firmly anchored within the society and developed as an essentially public good.

Acknowledgements. The authors would like to thank Antti-Juhani Filpus, Dmitri Fursov, Teemu Hakala, Jukka Korpihete, and Matti Viita for their constructive comments on various versions of this paper, as well as the founding members of the ELL-i open source co-operative for their initiative and courage.

REFERENCES

[1] Kemal Akkaya and Mohamed Younis. A survey on routing protocols for wireless sensor networks. *Ad hoc networks*, 3(3):325–349, 2005.
[2] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.
[3] Chris Anderson. *Makers: the new industrial revolution*. Random House, 2012.
[4] Kevin Ashton. That 'Internet of Things' thing. *RFiD Journal*, 22:97–114, 2009.
[5] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010.
[6] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *Int. J. Ad Hoc Ubiquitous Comput.*, 2(4):263–277, June 2007.
[7] Massimo Banzi, David A Mellis, David Cuartielles, Clay Shirky, Paul Badger, Tom Igoe, Federico Fissore, Roberto Guido, and Michael Shiloh. Arduino. The official Arduino web page at http://arduino.cc.
[8] Richard Barry. FreeRTOS. *Internet, Oct*, 2008.
[9] Adam Bennette. *Recommended Practice for DMX512: A Guide for Users and Installers*. United States Institute for Theatre Technology, 1994.
[10] Steven T Bushby. Bacnet: a standard communication infrastructure for intelligent buildings. *Automation in Construction*, 6(5):529–540, 1997.
[11] James Carter. *Raspberry Pi Power over Ethernet (PoE) Adapter by Xtronix Ltd*. Whitchurch Hill, Reading, Berks, United Kingdom, June 2013.
[12] Vint Cerf. The next 50 years of networking. In *the ACM97 Conference Proceedings*, 1997.
[13] Navraj Chohan, Chris Bunch, Sydney Pang, Chandra Krintz, Nagy Mostafa, Sunil Soman, and Rich Wolski. AppScale: Scalable and open appengine application development and deployment. In *Cloud Computing*, pages 57–70. Springer, 2010.
[14] Digital addressable lighting interface activity group. DALI Manual, Dali AG, ZVEI-Division Luminaires, 2001.
[15] Adam Dunkels. Full TCP/IP for 8-bit architectures. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 85–98. ACM, 2003.
[16] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki — a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462. IEEE, 2004.
[17] Adam Dunkels, Oliver Schmidt, Thiemo Voigt, and Muneeb Ali. Protothreads: simplifying event-driven programming of memory-constrained embedded systems. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 29–42. Acm, 2006.
[18] Eclipse foundation Machine-to-Machine industry working group – charter. Eclipse Foundation, Inc., Centrepointe Drive, Ottawa, Ontario, Canada.
[19] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol–http/1.1, 1999.
[20] Fadi Hamad, Leonid Smalov, Anne James, et al. Energy-aware security in m-commerce and the internet of things. *IETE Technical review*, 26(5):357, 2009.
[21] Introduction to Power over HDBaseT. HDBaseT Alliance, Beaverton, OR, USA, September 2011.

[22] Guido Hertel, Sven Niedner, and Stefanie Herrmann. Motivation of software developers in open source projects: an internet-based survey of contributors to the Linux kernel. *Research policy*, 32(7):1159–1177, 2003.
[23] Karl A Hribernik, Zied Ghrairi, Carl Hans, and K-D Thoben. Co-creating the Internet of Things–first experiences in the participatory design of intelligent products with Arduino. In *Concurrent Enterprising (ICE), 2011 17th International Conference on*, pages 1–9. IEEE, 2011.
[24] IEEE 802.3 standard for Ethernet. IEEE Computer Society, New York, NY, USA, December 2012.
[25] IEEE 802.3at standard for Ethernet. Amendment 3: Data terminal equipment (DTE) power via the media dependent interface (MDI) enhancements. IEEE Computer Society, New York, NY, USA, September 2009.
[26] IEC 14908 Open data communication in building automation, controls and building management - control network protocol. International Electrotechnical Commission, rue de Varembe, Geneva, Switzerland.
[27] ITU-T focus group on M2M service layer. International Telecommunication Union, Telecommunication Standardization Bureau, Place des Nations, Geneva, Switzerland.
[28] Trevor Johnson, Steve Meirowsky, Johnathon Weare, Ryan Walmsley, and et.al. Raspberry Pi foundation. Wikimedia Foundation, San Francisco, California, USA.
[29] Tim Kindberg, John Barton, Jeff Morgan, Gene Becker, Debbie Caswell, Philippe Debaty, Gita Gopal, Marcos Frid, Venky Krishnan, Howard Morris, et al. People, places, things: Web presence for the real world. *Mobile Networks and Applications*, 7(5):365–376, 2002.
[30] KNX standard. KNX Association, De Kleetlaan, Brussels/Diegem, Belgium, 2011.
[31] Venkat Kuppuswamy and Barry L Bayus. Crowdfunding creative ideas: the dynamics of projects backers in Kickstarter. Technical report, SSRN Working Paper, http://papers. ssrn. com/sol3/papers. cfm, 2013.
[32] Chris Lattner and Vikram Adve. LLVM: A compilation framework for lifelong program analysis & transformation. In *Code Generation and Optimization, 2004. CGO 2004. International Symposium on*, pages 75–86. IEEE, 2004.
[33] Gwendolyn K Lee and Robert E Cole. From a firm-based to a community-based model of knowledge creation: The case of the Linux kernel development. *Organization Science*, 14(6):633–649, 2003.
[34] Tomás Sánchez López, Damith C Ranasinghe, Mark Harrison, and Duncan McFarlane. Adding sense to the internet of things. *Personal and Ubiquitous Computing*, 16(3):291–308, 2012.
[35] Cristian Maglie. Arduino IDE 1.5 3rd party hardware specification.
[36] Sanjaya Maniktala. *Switching power supplies A to Z*. Elsevier, 2012.
[37] David A Mellis, David Cuartielles, Federico Fissore, and Alberto Cicchi. So you want to make an Arduino.
[38] *ENC28J60 Data Sheet – Stand-Alone Ethernet Controller with SPI Interface*. Microchip Technology Inc., West Chandler Blvd., Chandler, Arizona, USA, 2008.
[39] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516, 2012.
[40] Modbus application protocol specification v1.1b. Modbus Organization, Hopkinton, MA, USA, December 2006.
[41] Deborah Groban Olson and John Logue. Fix globalization: Make it more inclusive, democratic, accountable and sustainable. Capital Ownership Group, Ford Foundation, October 2002.
[42] Raspberry Pi. Rasberry Pi Foundation, Caldecote, Cambridgeshire, UK.
[43] Teemu Rinta-aho, Mika Karlstedt, and Madhav P Desai. The Click2NetFPGA toolchain. In *USENIX Annual Technical Conference*, 2012.
[44] Jean-Charles Rochet and Jean Tirole. Platform competition in two-sided markets. *Journal of the European Economic Association*, 1(4):990–1029, 2003.
[45] Diane Rowland, Carolyn DiGuiseppi, Ian Roberts, Katherine Curtis, Helen Roberts, Laura Ginnelly, Mark Sculpher, and Angela Wade. Prevalence of working smoke alarms in local authority inner city housing: randomised controlled trial. *Bmj*, 325(7371):998, 2002.
[46] Zach Shelby, Klaus Hartke, and Carsten Bormann. Constrained application protocol (CoAP). Technical report, Internet Engineering Task Force, 2013.
[47] *Arduino Ethernet Shield with PoE*. Spark Fun Electronics Inc., Longbow Drive, Boulder, Colorado, USA, 2013.
[48] *Arduino Ethernet with PoE*. Spark Fun Electronics Inc., Longbow Drive, Boulder, Colorado, USA, 2013.
[49] *POEthernet Shield*. Spark Fun Electronics Inc., Longbow Drive, Boulder, Colorado, USA, 2013.
[50] *STM32F05xxx/06xxx advanced ARM-based 32-bit MCUs, Reference Manual*, May 2013.
[51] Ovidiu Vermesan, Peter Friess, Patrick Guillemin, Sergio Gusmeroli, Harald Sundmaeker, Alessandro Bassi, Ignacio Soler Jubert, Margaretha Mazura, Mark Harrison, M Eisenhauer, et al. *Cluster SRA 2011*, chapter Internet of Things strategic research roadmap, pages 9–52. European Research Cluster on the Internet of Things, 2011.
[52] Zhong Lin Wang. Toward self-powered sensor networks. *Nano Today*, 5(6):512–514, 2010.
[53] Mark Weiser. The computer for the 21st century. *Scientific american*, 265(3):94–104, 1991.
[54] Geng Wu, Shilpa Talwar, Kerstin Johnsson, Nageen Himayat, and Kevin D Johnson. M2m: From mobile to embedded internet. *Communications Magazine, IEEE*, 49(4):36–43, 2011.

Edited by: Ethiopia Nigussie, Andreas Riener and Liang Guang
Received: Sep 1, 2013
Accepted: Sep 30, 2013

DOI 10.12694/scpe.v14i3.858
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 14, Number 3, pp. 169–179. http://www.scpe.org

P SYSTEM TESTING WITH PARALLEL SIMULATORS - A SURVEY

ALEX CIOBANU *AND FLORENTIN IPATE †

Abstract.
In the world of P Systems many different parallel simulators have been developed, both software and hardware based. There are software implementations with parallel computing technologies such as MPI and Hadoop as well as hardware implementations on FPGA and GPGPU. In parallel, significant theory has been developed in the world of testing P Systems based on execution paths. These testing algorithms require significant computational power to develop the oracles, a task well suited to the developed simulators. This paper will look at the different technologies used to develop simulators (and the simulators built with those technologies) and evaluate their adaptability for use in developing testing oracles, required for the already developed testing theory.

1. Introduction. Membrane computing, a field of study initiated by Gheorge Paun in [5] [12], has developed a computational model called P Systems which can very naturally model natural phenomena. As such, significant effort has been expended into developing simulators for P Systems as to allow for complex evaluation of these P Systems. Unfortunately given the inherent complexity of natural systems, complexity which P Systems share, simulators require significant computational effort in execution. To overcome these issues significant research has been done in leveraging parallel computing in the development of simulators. This includes attempts to use clustered computing as per the use of MPI (Message Passing Interface), as well as hardware attempts using GPGPU (General-Purpose Graphical Processing Units) and FPGA (Field Programmable gate arrays). As with any computer related system, testing is a must and significant testing theory has been developed as well. The P System community has developed several different ways of approaching testing theory at its core: the simple rule coverage and context dependent rule coverage testing. Possibly unsurprising, the attempts at scaling testing theory to non trivial P Systems have been limited by the ability to simulate these P Systems. A article was recently published which used Hadoop as the simulation engine for P Systems [6]. Unique in the approach was the explicit trailering of the simulator output for developing testing oracles; fundamentally this was an attempt at finding an intersection point between these two research fields. Based on that article we will now attempt to do a survey of other parallel simulator environments and the ways in which whey would be suitable / modifiable for use in P System testing.

For the purpose of P System testing there are two features a simulator must have above and beyond typical functionality as to be useful and those are the ability to do scoring and the ability of backtracking. Scoring is the ability to check off the discovered test sequences which were discovered throughout the simulation and backtracking is the ability to list all evolutions of the system from current state to initial state (basically defining a test oracle). Doing context dependent rule coverage also requires historical comparison, as to enable discovery of rule applications in sequential steps in the evolution of the P System. This three features, though banal in a non distributed environment, become quite difficult to achieve when using parallel system, as the parallelism can be interrupted by these requirements. We will bulk the different types of simulators into two groups (software and hardware) and example how these need to be modified as to enable scoring, backtracking, and historical comparison.

This paper does a survey of architectures of existing P System parallel simulators, and examines methodologies for using these simulators in P System testing. At the end of the article we look at the potential performance of such test systems.

This paper is structured as following: First we formally introduce P Systems, and offer an example of what a P System looks like. Next we do an architectural description of Software and Hardware based simulators, and explain currently developed testing theory. We then look at the architecture of augmented software and hardware simulators useful for P System testing. We examine what the interface between the P System testing tools and simulators would look like, and at the end give some perspective on scaling and performance possibilities.

*Department of Computer Science, Faculty of Mathematics and Computer Science, University of Pitesti,Str. Targu din Vale 1, 110040 Pitesti, Romania (alex.ciobanu@gmail.com).
†Department of Computer Science, Faculty of Mathematics and Computer Science, University of Bucharest,Str. Academiei nr.14, sector 1, Bucharest, Romania (florentin.ipate@ifsoft.ro).

2. Preliminaries.

2.1. P System. A (cell like) P system is a computational model inspired by the chemical reactions across cell membranes. Throughout the years many different types of P System have been developed with different types of membranes, varying rules structures and others. For our discussion the classical cell like P System with static structure will be used, though the theory presented could be extended to any other type of P System. The formal definition for a P System with which we will be working is:

DEFINITION 2.1.1. *A P system is a tuple*

$$\Pi = (\mathcal{V}, \mu, \mathcal{W}_1, \ldots, \mathcal{W}_n, \mathcal{R}_1 \ldots \mathcal{R}_n)$$

where

- $\mathcal{V}$ *is the alphabet (a finite and nonempty) set of objects*
- μ *is the membrane structure, a hierarchical arrangement of compartments named membranes identified by integers 1 to n. As notation, in this document the beginning and end of a compartment is denoted with a "[" and a "]" respectively. The compartment identifier is placed directly after the end "]" marked with a single quote.*
- $\mathcal{W}_i$ *where* $1 \leq i \leq n$ *are strings over* $\mathcal{V}$*, describing the multisets of objects initially placed in the i regions (compartment) of* μ*.*
- $\mathcal{R}_i$ $0 \leq i \leq n$ *is a finite set of evolution rules for each region i where evolution rule r is of the form*

$$r : u \to (a_1, t_1) \ldots (a_n, t_n) \tag{2.1}$$

where u and a_i *is a nonempty multiset over* $\mathcal{V}$*, and* t_i *is an element from* μ*.* t_i *is limited to the current membrane, the region immediately outside the current membrane or any of the membranes immediately inside the current membrane.*

The simulation of a P System begins with each membrane containing a multiset of objects described by the $\mathcal{W}_i$ variable in the definition. Next a set of rules is chosen to be applied in each compartment (membrane) of the P System. This set of rules does not have to be unique as P Systems may be non deterministic, but the rules must be chosen as to be maximal. Maximality requires that with a step (evolution) of the P system every single rule which can be applied (within a context) is applied. In other words given an multiset $\mathcal{W}$ and a set of rules $\mathcal{R}$ we will continue to choose rules from $\mathcal{R}$ to be applied until either $\mathcal{W}$ is $\emptyset$ or there are insufficient objects in $\mathcal{W}$ to apply any of the rules in $\mathcal{R}$. Once the rules are chosen they are applied in a parallel way, where the resultant objects are sent to their respective membranes. The process then repeats for a respecified number of evolutions or until some stop condition is reached [5].

To Exemplify consider we have the following P System:

$$\Pi = (\mathcal{V}, \mu, \mathcal{W}_1, \mathcal{R}_1),$$

where

- $\mathcal{V} = \{s, a, b, c\}$
- $\mu = []'1$
- $\mathcal{W}_1 = s^2$
- $\mathcal{R}_1 = \{$
  $r_1 : s \to (ab, 1)$
  $r_2 : s \to (a, 1)$
  $r_3 : a \to (b, 1)$
  $r_4 : a \to (c, 1)$
  $r_5 : b \to (bc, 1)\}$

If we were to perform simulations on the example P system we would start out with an initial configuration of $[s^2]'1$ then choose the rules to apply for the first step. In our case we can choose one of three options: r_1^2, r_2^2 or $r_1 r_2$, any other combination of rules would either require objects which are not available or would not fit the maximality principle. If we choose r_1^2 our resultant multiset would be $[a^2b^2]'1$ as we would apply the first rule twice consuming the two s in the membrane and creating two times ab. We can now choose again rules to apply and the process continues until a pre-defined number of steps occur or we run out of rules we can apply.

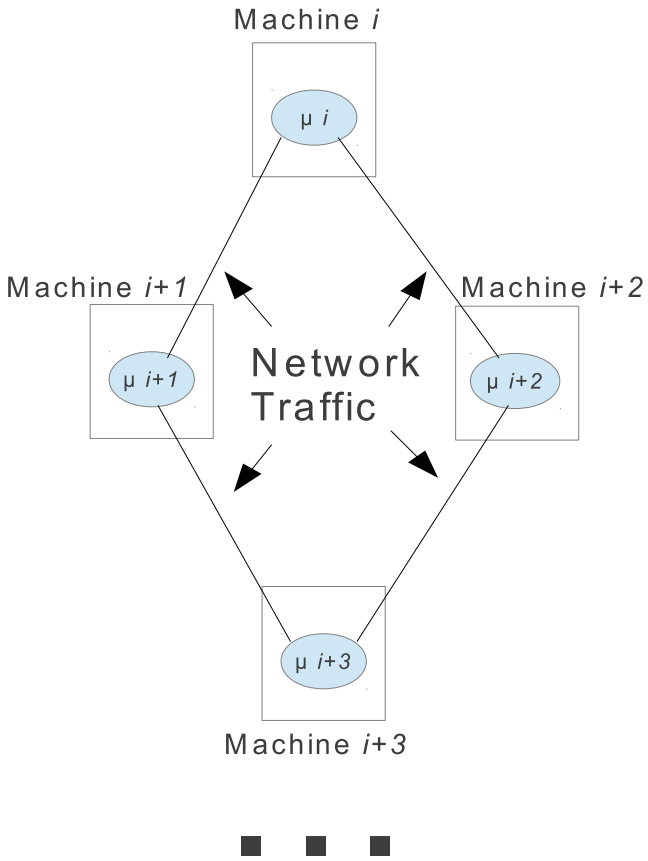

FIG. 2.1. *Software based simulator architecture*

2.2. Software Simulators. Though there is a wide variety of software based parallel P System simulators they all follow very similar architectural paradigm. This class of simulators models the distributed network of computers by the membrane structure, where each machine hosts one or more membranes of the P System. It is important to note, our running example in this paper uses a single membrane P system for simplicity of exemplification, but complex P systems (which would require such simulation architecture) usually have membranes which communicate with the different internal rules. Within these multi membrane P systems each membranes evolve independently on each machine though some coordination occurs when membrane communication is required. When particular rules of the P System require communication (the movement of one object from one membrane to another) the network connectivity between the servers is used to communicate the object movement. Some form of coordination is also used to ensure all membranes are on the same step of the evolution as to avoid inconsistent states. There are several variants on this particular theme well summarized in [2] using different programming languages C++, Java, and different approaches. There have also been attempts at increasing the effectiveness of the implementations such as [3] though the fundamental approach remained unchanged, hence our testing requirements would be identical regardless of optimizations. In Figure 2.1 we can see a pictographic representation the distribution model of membranes across multiple machines for simulation.

2.3. Hardware Simulators. In the world of P System Simulators there have also been several attempts at using computer hardware for example [1] (a comprehensive list can be found in [4]), for simulating P Systems, arguably the most scalable being the use of GPGPU processing power. This approach harnesses the power of computer video cards' computational power in simulating P Systems. The computational part of video cards is used to transform three dimensional data from a video game or CAD tool, into a displayable two dimensional image. Since most artefacts (elements to be displayed on the screen) can be computed independently, the processing unit of video cards essentially becomes a massively parallel computation infrastructure as to increase performance in rendering. This parallel computation of video cards has sparked the imagination of researchers as the number of FLOPS (Floating-point Operations Per Second) of a graphic card is orders of magnitude higher than standard general purpose processing units. Due to this trend, video card manufacturers have developed API libraries and architectural changes to better enable the use of graphic cards for general purpose parallel computing hence GPGPU (General Purpose Graphics Processing Unit).

As with the software simulators, several different approaches have been tried but they are usually variants on the same theme. The GPGPU hardware is designed to enable high number of simulations threads (usually in the hundreds) which all execute in parallel. In the P System simulators each membrane is placed in an independent thread. Depending on implementation and type of P System different approaches have been used to implement membrane communication where the final objects in each membrane are somehow encoded to the memory of the video card. Each evolution of the P System consists of another run through the video card pipeline. Though many different optimizations and executions styles exists for the purposes of this article this basic definition will suffice for our exploration. In figure 2.2 we represent photographically the distribution of membranes cross different parallel pipes with the GPGPU device.

2.4. P system Testing. A P System by its definition has different primitives compared to classical programming languages. In standard testing theory we deal with conditional statements, lines of code, looping constructs and other such artefacts. A P system does not contain any such elements, and is constructed from rules in membranes acting upon objects. As such testing theory was adjusted to deal within these artefacts when attempting to test P systems. Most of the theory developed focused on the rules application (within a membranes) as it is the driving force within a P system. Two different methods were developed in testing the rules of a P system, namely simple rules coverage, and context dependent rule coverage. Significant effort has gone into creating methodologies for generating the test oracles. Usually these use one or more of these categories: Grammar based methods, Finite state machine based methods, model checking, and derivation tree based methodologies [7]. Initially the most promising methodology was through the use of model checking software such as NuSMV [9] and SPIN [10]. The approach rewrites a P system as a Kripke structure representations, and uses a series of LTL (Linear Temporal Logic) formulas to extract test sets. This initial approach was very effective in developing test oracles but was quite limited in its ability to scale. SPIN was also used in a similar manner where the P System was implemented in Promela (the modelling language of SPIN) and much better results were achieved, though this approach was limited to the capabilities of a single machine. The combination of existing theory of X-machines testing theory and P Systems was also attempted [11] which also gave methods of formally testing P Systems. All of these approaches though very innovative, were at times limited in their ability to leverage existing high powered parallel simulators such as those mentioned in the introduction. The technique which is possibly the most adequate for leverage existing hight powered P System simulators is the computation tree mechanism. Before we look at the different testing techniques we will first example the computation tree (the framework used in developing test oracle).

2.4.1. Computation Tree. In essence a computation tree is a tree with a root node as the initial configuration of the P System and each branch of the computation tree represents a possible application of rules. Each edge represents a possible rule application and the resultant configuration of the P System being the resultant node in the tree. All possible evolutions of the P System is represented as a branches from a particular node in the tree. The complete tree representing all possible configurations a P System can take, and the sum of all branches represents all possible evolutions. An example of a computation tree can be found in Figure 2.3. It is important to note that a computation tree can grow to be very large, a property that is determined by the degree of non determinism. The more possible paths available from a node, the higher the exponential growth of the tree in consequent levels. This non-determinism is inherent in the way P System are defined, a

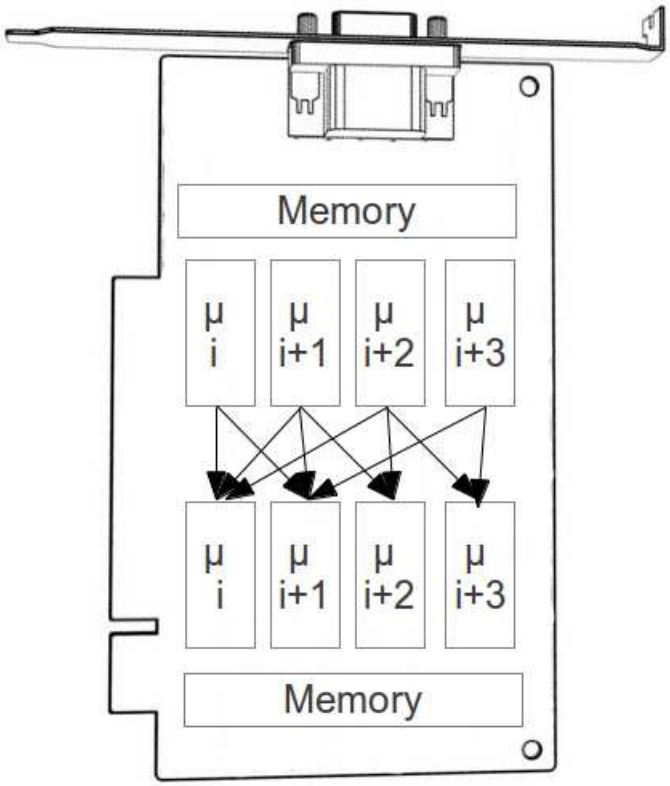

FIG. 2.2. *GPU based simulator architecture*

property that enable very natural modelling of physical systems. At the same time this explosion in the size of a derivation tree is also the motivating factor for using high performance computing (the parallel simulators) in dealing the non trivially sized P Systems.

2.4.2. Simple Rule Coverage. Simple Rule Coverage [9] consists of verifying the application of each individual rule in the P System. To have complete simple rule coverage every single rule in every membrane must be applied at least once, with the entire evolution of the P system. To develop a test oracle there are several methodologies including using formal verifiers, though for the propose of this article the most applicable method uses computation trees. Traversing the computation tree (in either a breath or depth first manner) would enable the extraction of a test oracle as the tree contains both the rule application and the resultant multisets. To offer an example we have the P System given in the preliminaries section. Simple rule coverage would require the application of all rules from r_1 to r_5 at least once, hence the following executions of the P System would offer complete simple rule coverage. The test vectors are displayed in the following format: multiset { with these rules applied }. A $\rightarrow$ is used to define the resultant configuration after applying the defined rules.

$$s^2\{r_1r_2\} \rightarrow a^2b\{r_3r_4r_5\} \rightarrow b^2c^2$$

Very important to note this is not the only test vector possible as the two test sequences below are equally

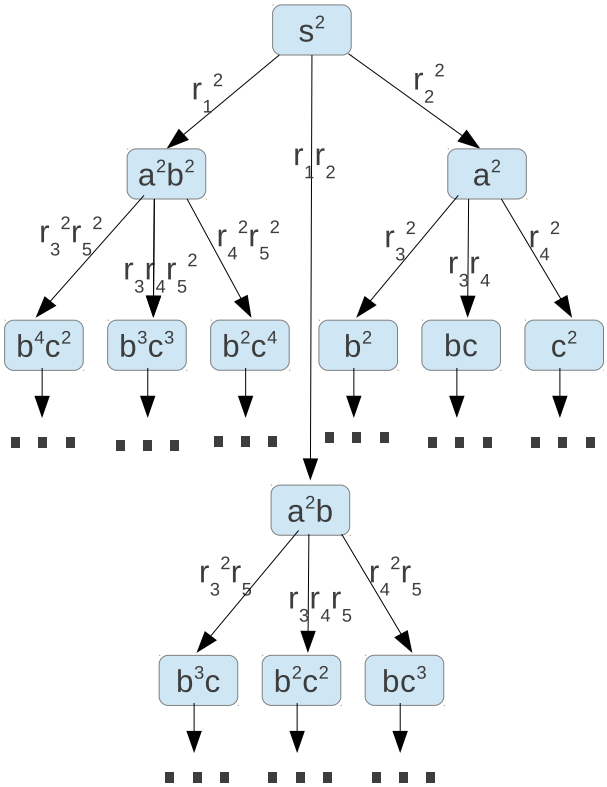

FIG. 2.3. *P system Computation Tree*

viable, though may take more time to execute as both test vectors must be executed to achieve simple rule coverage.

$$s^2\{r_2^2\} \to a^2\{r_4^2\} \to c^2$$
$$s^2\{r_1^2\} \to a^2b^2\{r_3^2, r_5^2\} \to b^4c^2$$

2.4.3. Context Dependent Rule Coverage. Context Dependent Rule Coverage [9] is a more complete test mechanism for P Systems, as it requires the testing of all possible rules in every possible context. Usually sequential steps in the simulation of P Systems would have some rules applied in sequential steps. For example in step 1 rule 1 is applied and in step 2 rule 3 is applied. Rules (1,3) would be a context dependent rule coverage pair which would need to be tested. All possible context dependent rule coverage pair them must be traversed by a simulator to achieve complete context dependent rule coverage. Context dependent rule pairs can be calculated without simulation of the P System, directly from the membrane and rule structure, and the test oracle again can be discovered from traversing a computation tree. For a discovery perspective this type of testing is much more difficult as it requires historical context and can not be calculated on the fly as simple rule coverage can. Context dependent rule coverage requires a system by which historical data is store to compare to current state. It is the comparison to historical data which enables use to determine context and is the most difficult element to augment on P system simulators. We can again look at an example of finding context dependent rule coverage test sets for the example P system in the preliminaries. If we look at r_1 and r_2 we see there is not rule which produces the s object required to apply those rules, hence there is no context dependent rule coverage pairs stating with those rules. If we look at r_3 which requires an a object to execute

we see that both r_1 and r_2 can generate this context, hence we have context dependent pairs (r_1, r_3), (r_2, r_3). We continue this process with the rest of the rules and discover additional rule pairs: (r_1, r_4), (r_2, r_4), (r_1, r_5), (r_3, r_5). We can do walks of the computation tree to discover the following test vector which would guarantee context dependent rule coverage testing.

$$s^2\{r_1, r_2\} \to a^2b\{r_3, r_4, r_5\} \to b^2c^2\{r_5^2\} \to b^2c^4$$

We can see by the more complicated test set required that Context Dependent Rule Coverage is a more stringent rule coverage criteria. It is important to note that although only consecutive rows (steps) in the computation tree are required to discover context dependent rule pairs, the entire tree is required when extracting the test oracle. A test oracle is required to start from the initial configuration of the system, hence a path (in the computation tree) from the currently discovered context dependent rule pair to the root note needs to be maintained. As there is no way of determining which paths will be useful until they are explored the full computation tree mush be maintain until all test oracle are discovered.

3. Augmenting Software Simulators for Testing. When looking at software simulators we see that most, if not all implementations, do not have provisions for neither scoring nor backtracking for one very simple reason. Both scoring and backtracking require the storage of at least meta data describing the evolution process and the cost (CPU and memory) of storing the information would impede performance, hence such data is not stored. For the use of these simulators in a testing context this storage of data is no longer optional and a method needs to be found which would impact least the performance of the system. Most critical for performance is the networking, as it has already been identified and a bottleneck for this type of simulators, hence any augmentations should be designed to produce the minimum amount of network traffic.

For data storage the standard would be some type of database technology but a typical RDBMS would require all data from each server in the simulation cluster to the shipped to central database which would put significant pressure on the already overburdened network. The use of local storage where each server in the simulation cluster stores data / metadata locally would be a great solution for the networking bottleneck but queering this data in a holistic manner would then be difficult as there is no link between the individual server's data. The most suitable solution would be a hybrid approach potentially a NoSQL database.

NoSQL databases are a new breed of database technologies coming out of the Web 2.0 sphere which take advantage of distributed computing to scale in both data size and simultaneous transactions. These databases usually run in distributed mode where the storage of multiple physical servers is used simultaneously as a persistence layer. For more information please read the description in [8]. This database would be installed on every single server in the simulation cluster, which would allow for data storage operations to execute directly on the same server. At the same time when cross membranes queries are required the database can perform network based queries where all of the servers are interrogated and a holistic view is provided.

The process of using the distributed clusters for the discovery would be as followed:

1. Load the test sets which need to be discovered into the NoSQL database, ensure each local storage server has information pertinent to which will be simulated on that server.
2. Begin the simulation process on all of the servers, where a small background process scores the discovered test scenarios.
3. Stop the simulator when all off the test scenarios have been discovered on all of the servers.
4. Perform complex queries on the NoSQL database to extract the test oracles based on all of the data stored on each of the servers.

Figure 3.1 has a pictographic description of the architecture which could be used.

4. Augmenting Hardware Simulators for Testing. The hardware simulators which have been developed use a slightly different paradigm for parallelism. The hardware used for these simulators is inherently parallel hence there is no need for chaining multiple devices together to achieve parallelism. The architecture of the device poses a completely different set of problems. Graphic cards are designed as high power matrix multipliers architected to rapidly compute output pixels based on incoming vectors and output these pixels to the graphics port. Video cards are not optimized for return data to the calling system and the calculation engine is not well optimized for branching or conditional instructions. This poses a significant issue when adapting the simulator for testing.

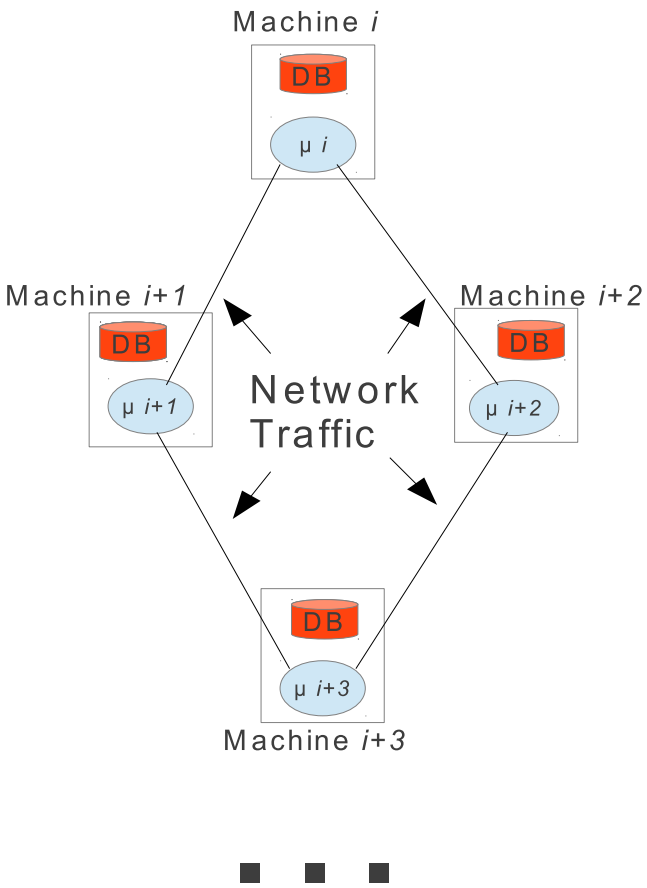

FIG. 3.1. *Software based simulator architecture which would be usable for testing*

There are two possible ways by which to augment the simulator for software testing.

1. Implement the testing requirements (backtracking and scoring) right on the hardware device itself
2. Move the results of the hardware simulator to the host system where they can be analysed for the required test oracles

The poorly optimized branch instruction would make approach 1 significantly impede the performance of the simulators hence approach 2 would be preferred. To reduce the impact of the low bandwidth return pipe, DMA (Direct Memory Access) interface to the GPU memory can be used. This will enable the fastest possible retrieval of data from the graphical device while reducing the impact on the CPU. The graphic card memory is usually used as a ring buffer, a feature which can be exploited for the data movement. Data is written by the simulator to memory and a read pointer pulls data to the main system. The read pointer can be far behind the write pointer without impact on data consistency as long as unread data is not overwritten. This will enable a certain amount of latency in reading while still allowing the simulator to run at full speed.

Once the data is pulled to the host system (possibly into a database), the same algorithms used for the software based simulators to extract test vectors and their oracles can be used. Given that data is moved from the video card into a database, it is theoretically possible to use the database as a synchronization service for chaining multiple GPUs together and create a GPU compute cluster. This is possibly to be explored in a future article.

This type of architecture can be used not only for GPU based simulators but for any hardware based

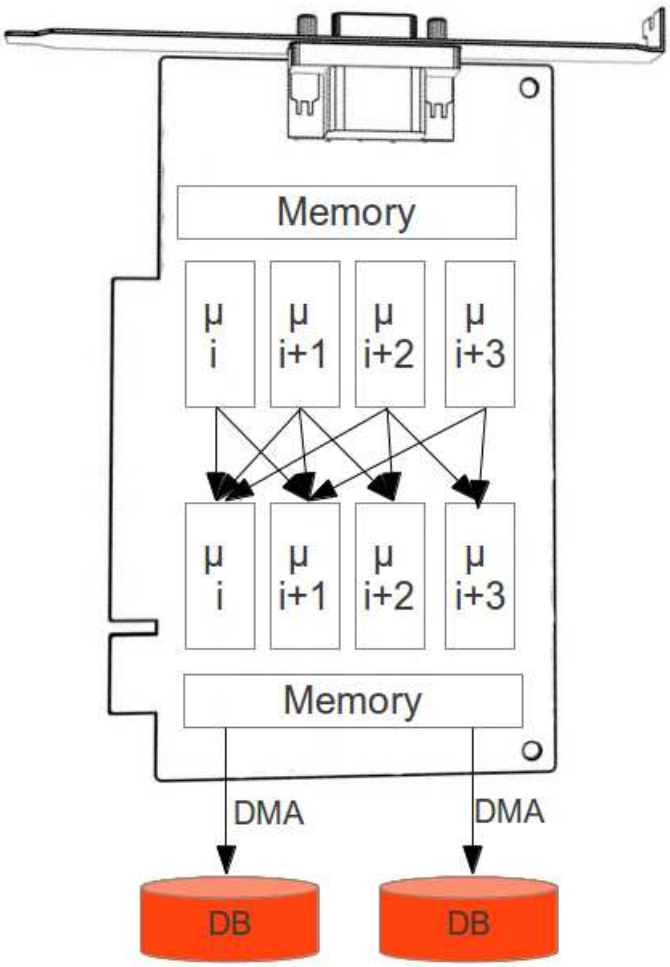

FIG. 4.1. *GPU based simulator architecture which would be usable for testing*

simulator though a different approach would need to be used for extracting data from the memory of the system. This extraction of data will be based on the type of hardware used and possible interface options. We can see in Figure 4.1 a conceptual diagram of what the architecture of such a system would look like.

5. Interface. When examining the different types of simulators both software and hardware, although the architectures vastly differ, they suffer from exact same drawbacks in their utility for P system testing: the lack of memory. Fundamentally but augmenting these systems with the ability to remember past events (previous steps in the simulation) all of these platforms would become idea engines for a testing apparatus. We have already shown in a previous article [6] how a NoSQL database can be used as a persistentcy technology providing the memory require for a simulator to become a testing tool. The NoSQL database greatly out scales the memory available on a video card, as well as the RAM available in test system used in a software simulators. We can also see from benchmarks [8] how a NoSQL database can provide the necessary inserts per second as to not become the bottleneck in running the simulator. Unfortunately in the previous article the NoSQL was explicitly integrated in the simulator providing a tight coupling between the two technologies and an inflexibility in using other simulators. It would be ideal to decouple the Persistence technology from the simulator as to create a generic pluggable component which can be accessed as simple generic API and store the P System computation

tree for any general purpose simulator. The interface should contain the following interfaces as to enable easy hook-in for established simulators.

1. Store Configuration possibly with ID for easy identification (Configuration ID)
2. Add Edge from Configuration (Configuration ID, Edge)
3. Add Target Configuration (Source ID, edge, Target ID)
4. Get all Configurations at a level (level number)
5. Get root configuration
6. Get all child configurations (Configuration ID)
7. Get parent configuration (Configuration ID)

Using these very primitive functions a execution tree can be created using any simulator which can call this interface. As well walks of the tree can be performed to compute test vectors and test oracles for simple rule coverage and context dependent rule coverage.

6. Preliminary Results. To see what scaling possibilities are available with this architecture we would need to look at some testing results of an augmented P System simulator. Although a fully generic NoSQL database interface which can snap onto any simulator is still in development, we do have results of a NoSQL database attached to a Hadoop based simulator of P Systems. The tests below used a very simple P system designed to allow for maximal use of the parallel architecture of Hadoop. The test looked at how many nodes in a computation tree could be added into the database, at each level of the tree.

Tree Level	Number Of Nodes	Duration
2	14	24s
3	196	26s
4	2 744	36s
5	38 416	48s
6	537 824	4m 13s
7	7 529 536	54m 18s

The cluster used for these experiments was composed of 12 machines each with a simple core 2 duo processor 4 GB of RAM and single SATA hard disk, with simple 100 MB unmanaged network connecting the machines. It is also important to note the time gating factor was the simulator and not the database, but these results go give an idea of scaling possibilities.

To see even large computation trees we can look at the article [6] where even larger computation tree was developed. In one experiment done for the article it was attempted to create a very large computation tree. These results used the same set-up as the previous experiments though there were 16 machines in the cluster and membrane communication was enabled. The number of nodes created in an experiment was:

Number Of Nodes	Storage Space	Executon Time
65 471 486	84.9 GB	16.54 hrs
77 186 334	105.4 GB	above + 5.52 hrs

This execution was again quite small compared to the possibilities of scaling. To get an idea of what a NoSQL database can do, we can look at published performance results for Oracle's NoSQl Database. There we see close to 100, 000 inserts per second into a 12 node cluster (though the nodes were Dual Xeon multi hard disk systems) [8]. The scaling results are also very high, possibly into the petabyes of data architecture.

7. Conclusion. In this article we have looked at several different architectures for parallel simulators of P Systems and their possible extensibility for use in P System testing. We looked at possible extending software simulators with NoSQL databases to allow for testing oracles to be extracted. We also looked at ways to extending hardware simulators (for use in P System testing) by again putting simulator data into a database on the host system. This article has looked at possible architectural configurations. It would now be interesting to attempt implementation of these architectures to measure potential benchmarks. There currently

already exists an implementation which uses Hadoop to generate a derivation tree which is stored in a NoSQL database, derivation trees which can be hundreds of millions of nodes in size. Further as part of the same project distributed architecture is used to walk the computation tree and discover the P System test oracles. Future work will now concentrate on extending the developed application to a more generic in interface, as to allow any simulator to insert into the NoSQL database and inherently build a computation tree. This will allow for explicit testing of all architectures described in this article and allow for a quantitative comparison of the different schemas. Further we will also look at possible hybrid implantations of the simulators mentioned in this article to find the most viable solution for scaling to large P System in our simulation and testing effort.

Acknowledgement. This work was partially supported by a grant of the Romanian National Authority for Scientific Research, CNCS–UEFISCDI, project number PN-II-ID-PCE-2011-3-0688.

REFERENCES

[1] L. Ferndez, V. J. Martatinez, F. Arroyo, and L. F. Mingo. A hardware circuit for selecting active rules in transition p systems. In SYNASC, pages 415-418, 2005.
[2] G. Ciobanu and G. Wenyuan. A P system running on a cluster of comput- ers. In Lecture Notes in Computer Science, WMC 2003, pages 123-150. Springer-Verlag, 2004.
[3] A. Syropoulos, L. Mamatas, P. C. Allilomes, and K. T. Sotiriades. A dis- tributed simulation of transition P systems. In Workshop on Membrane Computing, pages 357-368, 2003.
[4] M. A. Martinez-del-Amor: Accelerating Membrane Systems Simulators using High Performance Computing with GPU. PHD thesis University of Seville (2013).
[5] G. Păun: Computing with membranes. Journal of Computer and System Sciences 61(1), 108-143 (2000).
[6] A. Ciobanu, F. Ipate Using Big Data technologies with P systems CMC14 proceedings pages 95-116.
[7] M. Gheorghe, F. Ipate. Testing based on P systems - an overview. Proceedings of the Eleventh International Conference on Membrane Computing (CMC11), Jena, Germany, August 2010, pp. 7-10, ProBusiness Verlag, Berlin. Lecture Notes in Computer Science, vol. 6501, pp. 3-6, Springer, 2010.
[8] https://blogs.oracle.com/charlesLamb/entry/oracle_nosql_database_performance_tests/.
[9] F. Ipate, M. Gheorghe, R. Lefticaru. Test generation from P systems using model checking. Journal of Logic and Algebraic Programming, 79(6): 350-362. Elsevier Science Inc, 2010.
[10] F. Ipate, R. Lefticaru, C. Tudose. Formal verification of P systems using SPIN. International Journal of Foundations of Computer Science Vol. 22, Nr. 1/2011.
[11] T. Balanescu, M. Gheorghe, F. Ipate. Combined power of X-machines and P systems. Analele Universitii Bucureti, Informatic, Vol. LVIII, pp. 35-48, 2009.
[12] G. Păun, G. Rozenberg, A. Salomaa : The Oxford Handbook of Membrane Computing. Oxford University Press (2010).

Edited by: Dana Petcu
Received: Sep 1, 2013
Accepted: Oct 4, 2013

DOI 10.12694/scpe.v14i3.859
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 14, Number 3, pp. 181–187. http://www.scpe.org

JOIN PATTERNS FOR CONCURRENT AGENTS (POSITION PAPER)

ALEX MUSCAR*

Abstract. Even though the Agent Oriented Programming (AOP) paradigm has been introduced over 20 years ago, the following is still modest. We believe this is partially due to the lack of the community to address such real world issues as concurrency and distribution.

In this paper we take a look at the possibility of combining elements from the Join Calculus, namely *join patterns* with an AOP programming language, in our case Jason. We discuss some of the implications, advantages and possible drawbacks.

Key words: asynchronous programming, agent-oriented programming, concurrency

1. Introduction. Since the introduction of *Agent Oriented Programming* (AOP) two decades ago [14] the world of software has evolved considerably. The uprise of mobile and distributed applications has made the complexity of software increase exponentially. Another important change is that the number of cores in the modern processors is increasing steadily, giving rise to new opportunities. In the meantime, the AOP community has made little progress in making the agent paradigm a viable solutions of these kinds of problems. We believe that this is a cultural issue: the AOP community has been closed to external influences, thus it has not been able to take advantage of research efforts in other fields. This has started to change recently though, with efforts to make agent languages more oriented towards real world problems [12]—the authors propose a new agent language called SimpAL which addresses some shortcomings of existing agent languages such as weak type systems. The need for better agent languages is also clear from experience reports such as [11], which describes the experience of using Jason[1] [4], one of the best known agent languages, for developing a medium software project, and the problems faced by the author. Another relevant and recent effort in the area on adapting agent languages to real world issues has been presented in [6]—the authors have ported the Jason language to run on the Erlang virtual machine, a highly scalable and concurrency oriented platform. For an in-depth review of agent languages we refer the reader to [1].

In this paper we propose to take a look at how the *Join Calculus* [7], a process calculus for distributed systems, can be integrated with AOP languages, namely with Jason. More specifically, we propose to integrate *join patterns* [7] in the *triggering events* of Jason plans (see sec. 2). We have chosen Jason because it is one of the most popular and mature AOP languages. As of the Join Calculus, we have chose it since we feel join patterns integrate well with Jason. The author of [9] shows how the Join Calculus relates to *Constraing Handline Rules* (CHR), another language from the logic programming language family. Elements from the Join Calculus have also been incorporated in research branches of popular languages such as C# [2], leading to interesting results.

The rest of the paper is structured as follows: in sections 2, and 3 we make short overviews of Jason and the Join Calculus. In section 3.1 we introduce the proposed approach together with a working example. We extend the discussion in section 4. In section 5 we talk about some problematic aspects of the proposed approach. We finally conclude the paper in section 6.

2. An Overview of Jason. In order to make the code samples in this paper easier to understand, in this section we are going to give some details on the syntax of Jason. We will also very briefly and informally look at the Jason semantics.

2.1. The Syntax of Jason. Jason language has the notions of plans, beliefs, and goals. A program is made out of three sections:

- **Beliefs** are very similar to facts in Prolog [5] with the operator `&` replacing `,` in conjunctions. The belief base can be manipulated with the aid of the `+` and `-` operators, in the *body* of the plan. They

*University of Craiova, Bvd. Decebal, Nr. 107, 200440, Craiova, Romania (muscar@gmail.com). This work was partially supported by the strategic grant POSDRU/CPP107/DMI1.5/S/78421, Project ID 78421 (2010), co-financed by the European Social Fund - Investing in People, within the Sectoral Operational Programme Human Resources Development 2007 - 2013.

[1] http://jason.sourceforge.net/

add, and delete respectively, a belief from the belief base. Since these operations usually come in pairs, a deletion followed by an addition, the shortcut operator `-+` is used for this purpose;
- **Goals** are introduced by the `!` operator[2];
- **Plans** which follow the generic form `triggering_event : context <- body.`, are intended to handle goals. *Triggering events* match goals and message. The only relevant triggering events for this paper are goal addition and message arrival, denoted by atoms with the `+!` and the `+` prefixes.

Internal actions allow agents to call methods from the underlying platform. They are qualified by their package name (like in Java, e.g. `package.action`). Predefined actions do not have a package name, but they retain the leading period (e.g. `.send`). For further details we direct the interested readers to [4] for further details.

2.2. Informal Jason Semantics. Jason uses the concept of *beliefs*, which represent the agent's view of itself and the world, *goals*, which represent states in which the agent wants to be, and *plans*, which are "recipes" for action. An agent is driven forward by a *reasoning cycle* in which the agent: (i) perceives the environment, and updates its beliefs, (ii) receives messages, (iii) selects an event and the plans that can handle it, and (iv) executes (a step of) the plan. Note that running plans are organised as *intentions*, which are stacks of currently executing plans. The Jason interpreter uses a single OS thread for executing all the intentions, thus a single intention is active at any given time. While this is a (very) simplified description of the reasoning cycle, it has all the information that we need.

3. Combining Join Calculus and Practical Reasoning Agents. In this section we take a brief look at the Join Calculus and how join patterns can be integrated in Jason in order to allow intra-agent concurrency.

The core Join Calculus is similar to a call-by-value lambda calculus enriched with concurrency primitives [7]: `run`, which starts a concurrent evaluation of an expression in a new *process*; the | operator for parallel composition of processes; and *join patterns* which serve for process synchronisation. Processes are represented as unique names with arguments, akin to compound terms in Prolog.

Since the Join Calculus is based on the *Chemical Abstract Machine* (CHAM) [3], its semantics is defined in terms of *reaction rules*. In the case of the Join Calculus, a reaction rule is composed from a join pattern and a join body, and has the general form $J \triangleright B$, where J is the join pattern, and B is the join body. A join pattern is constructed from a list of processes. The semantics of a join pattern is as follows: whenever all the processes in a join pattern are active in the program, they are consumed, and the processes in the join body are added to the set of active processes. For an example see sec. 3.1.

For an in-depth overview we refer the reader to [7]. Also note that this is only the core of Join Calculus. The following sections will refer to the richer flavour of Join Calculus implemented by the JoCaml system [10].

3.1. Join Patterns. *Join patterns* are a declarative way of defining synchronisation patterns between *channels* (or *ports*). Listing 1 shows the implementation of an unbounded buffer using a join pattern [7].

```
put(x) & get() = reply x to get
```

Listing 1: A simple unbounded buffer implemented using join patterns

The succinctness of the code is due to the semantics of the join patterns. As mentioned in sec. 3, in order to *activate* the pattern, messages must be sent to on both channels. When a pattern is activated its body is executed. In a scenario where multiple producers would send values on the `put` port none of them would block, because `put` is *asynchronous*. If a consumer were to call `get` and and no prior `put` message was sent to the buffer, then the caller would *block* until such a message is sent. This is because `get` is a blocking (or *synchronous*) channel. This is specified by returning a value on this port in the body of the pattern. When a join pattern is activated, the values that triggered the pattern are *consumed*. This is an important issue to which we will return later.

[2]To be precise the `!` introduces *achievement* goals. Jason also has *test* goals introduced by the `?` operator, but they are not yet supported by our dialect yet, and as such relevant for the purpose of this paper.

3.2. Declarative Beliefs and Goals. As mentioned earlier, Jason features declarative beliefs and goals. Listing 2 show a part of a coffee maker agent written in Jason. While not complete, the listing is enough to show the flavour of developing agents in Jason.

4. Declarative Concurrency for Agents. We believe that by combining the declarative approach of specifying agents by using knowledge, beliefs, and goals on one side and join patterns on the other we can get a stronger language that maintains its declarative nature, but which also gains declarative concurrency. Assuming a mental state similar to the one in listing 2[3], listing 3 shows the implementation of a coffee maker agent in Jason extended with join patterns.

We will now take an in-depth look at this version of the coffee maker agent. While in Jason the triggering events use predicates, Jason extended with join patterns uses a mixture of channels and predicates. In listing 3, channels are written with a bold font face, while predicates are written with a regular typeface. Some readers may already be wondering what happens with variables during the resolution process? How do they unify?

When a pattern is activated, the variables in the synchronous channel(s) are bound to the values that activated the pattern. The variables in the asynchronous channels are left unbound and they get values by substitution during the resolution process. For *every* possible substitution a test is made to see if the substitution matches values present in the channel—not necessarily the latest value, channels are scanned for the required value. When a match is found, the matching values are consumed, the free variables are bound, and the body of the pattern is executed. If there is no match, or the whole pattern can not be grounded the value in the synchronous channels is put back in the buffer and the body of the pattern does not execute.

Algorithm 1 shows the algorithm defined above. The channels are extracted to form a join pattern, $\{c\} \cup A_c$—c is the first channel in the pattern—while the predicates are extracted to form a linear sequence of clauses, $C = C_1, C_2, \ldots, C_n$. When the pattern is activated the mental state of the agent is used to find the *unifier* σ. Then we scan (i.e. look through the values in a channel's message queue without consuming them) the remaining asynchronous channels, using the SCAN method, to see if we can find the values for the unifying substitution, using the VAR function to extract the variables from the channel. If we find matches for every channel, then we execute the body, b, of the action pattern to which we apply σ. In the opposite case, we re-queue the value that triggered the computation in the synchronous channel, which we obtain by using the VAL function, so that it can be used in future matches.

There are a couple of restrictions that we impose on our model:

- There must be *at most one* synchronous channel in a pattern (note that there can be patterns made only out of asynchronous channels); and
- Before a pattern can be activated all the variables in it must have a value.

There might be multiple *activable* patters at any moment. The method we presented would treat each pattern in turn, from top to bottom, using the previously illustrated algorithm to decide which pattern can be activated. Once a pattern is activated no other pattern will be considered in that step. This is similar to how Prolog defines predicates out of multiple clauses or to how functions are defined in Haskell, by defining multiple equations with the same name. In our case, the blocking channel can be considered the central part.

When we said, above, that the mental state of agent enhanced with join patterns is not exactly the same as that of the Jason agent we were referring to the fact that beliefs in the proposed language are represented as channels, rather than predicate. This has advantages as well as some drawbacks. The main advantage of this approach is that we have a uniform model, where the beliefs can be simply updated by sending messages, instead of receiving a message and duplicating the information in the belief base. The main drawback is that once a patter in which a belief takes part is fired, the value of the belief is lost. In the case of depleatable resources, such as water or coffee beans in the case of the coffee maker agent, this acts in our favour since we don't need to manually manage the resource (first add it, then retract it). But for more long-lasting beliefs this is undesirable. If we wish to keep this representation model, we need to find a way around this issue. We have come up with two very similar approaches:

- *Annotations* introducing metadata in the language (e.g. Java annotations or C# attributes). This could prove beneficial in the long run since this is a general approach which could be used in other cases

[3]This is not entirely the case, we will come back to this issue later

```
// Rules

neededFor(coffee, water).
neededFor(coffee, grounds).
neededFor(grounds, grounds, beans).

canMakeIt(M, P) :-
  canMake(M, Prods) &
  member(P, Prods).

// Beliefs

have(water).
have(beans).

canMake(maker, [coffee]).

// Initial goals

!have(coffee).

+!have(P)
<- // If we want to have some product P
   // then make it.
   !make(P).

...

+!make(P)
: // If we need R in order to make P
  neededFor(P, R) &
  // and we don't have it
  not(have(R)) &
  // but we can make it ourselves
  .my_name(Me) & canMakeIt(Me, R)
<- // then we just set a goal to have R
   !have(R).

+!make(P)
: // If we need R in order to make P
  neededFor(P, R) &
  // and we don't have it
  not(have(R)) &
  // but we know someone else who can
  // make it
  .my_name(Me) & canMakeIt(Maker, R) &
  Me \== Maker
<- // the we just tell that agent to make it
   .send(Maker, achieve, have(R)).

...
```

Listing 2: A coffee maker agent

```
Make(X) & neededFor(X, Y) & have(Y)
<- .reply(Make, X).
Make(X) & neededFor(X, Y)  & neededFor(Y, Z) &
  have(Z)
<- Make(Z).
Make(X) & neededFor(X, Y) & canMake(Producer, Y)
<- .send(Make, have(Y)).
```

Listing 3: A coffee maker agent in Jason with join patterns

Algorithm 1 Jason with join patterns pattern activation algorithm

procedure ACTIVATE(c, A_c, C, b)
  $\sigma \leftarrow$ UNIFY(C, *metal_state*)
  activate $\leftarrow True$
  for c_a in A_c **do**
    value $\leftarrow$ VAR(c_a)σ
    if $\neg$ SCAN(c_a, *value*) **then**
      activate $\leftarrow False$
    end if
  end for
  if *activate* **then**
    ($b\sigma$)()
  else
    $c \leftarrow$ VAL(c)
  end if
end procedure

as well. In this case an annotation `@keep` could be introduced to mark a channel whose value is to be kept even after a pattern has been activated; and

- *Special-purpose predicates* in the spirit of `bel` or `goal` from the GOAL language [8].

Irrespective of the design decision in this case a mechanism for manually consuming the value from a channel is needed. Listing 4 illustrates this issue. When an agent enters in a new environment (signaled by the runtime via the channel `EnterEnv`) it starts believing that it is in that environment (by means of the `inEnv` channel/belief in our example). When, later, the agent needs to know in which environment it's operating in and it queries its `inEnv` channel/belief, if the pattern matches the value of the environment will be consumed. This is clearly undesirable.

```
EnterEnv(Env) & canMake(X)
<- .send(Env, register(canMake(X))), inEnv(Env).

...

Make(X) & neededFor(X, Y) & inEnv(Env)
<- // Find out who can produce Y
   .send(Env, ask, query(canMake(Producer, Y)));
   // Start believing that Producer can make Y
   +canMake(Producer, Y)

...
```

Listing 4: Unintentionally consuming values

In listing 5 we illustrate both approaches outlined above. For the above mentioned reasons we favor the annotations approach.

```
...

// Specifying that a value is to be
// kept in the channel via annotations

Make(X) & neededFor(X, Y) & @keep inEnv(Env)
<-
// Find out who can produce Y
  .send(Env, ask, query(canMake(Producer, Y)));
// Start believing that Producer can make Y
  +canMake(Producer, Y)

// The same effect via special-purpose
// predicates

Make(X) & neededFor(X, Y) & keep(inEnv(Env))
<-
// Find out who can produce Y
  .send(Env, query(canMake(Producer, Y)));
// Start believing that Producer can make Y
  +canMake(Producer, Y)

...
```

Listing 5: Keeping values unconsumed

5. Open Issues. Besides the belief representation issue, which is a fundamental design decision for the language, there are a series of other open issues/design decisions:

- **Goal representation** Similarly to the belief representation issue, we have the goal representation issue. Again, we have two options: i) keep the predicate representation that Jason uses and introduce inconsistency in the language—different mechanisms are used to represent beliefs and goals; or ii) use channels, just like in the case of beliefs. Maybe a combination of the two is possible: since a goal is a desired state of the world, and an agent's view of the world is modeled by its beliefs it seems natural to represent goals as patterns of channels and predicates (just like action patterns). This is an important design decision and its effects need to be carefully considered;
- **Channel visibility** In the previous examples we have operated under the assumption that the capitalization of the channel names dictates their visibility: upper-cased channel names represent public channels, while lower-case names denote private channels;
- **Advertising capabilities** In listing 4 the agent exposed its capabilities by directly advertising them to the environment (i.e. `.send(Env, register(canMake(X)))`). While this works for a toy example it can get unwieldy to expose capabilities via this mechanism. One alternative would be to look at agents as abstract types that conform to an interface (like in the ML family of languages), and register that interface with the environment;
- **Communication** So far we haven't said anything about communication. What is the level at which agents communicate? In our previous examples agents can communicate at the knowledge level, by freely exchanging compound terms (functors). It is still not clear if this is the best approach, though it seems like it; and
- **Predicates in the body of a pattern** In listing 4 we used `.send(Env, query(canMake(Producer, Y)))` to find out which agent can make a certain item. While this is elegant, it's not entirely clear where the `Producer` variable gets bound (i.e. the semantics of the distributed/remote resolution semantics are not clear).

6. Conclusion and Future Work. Our approach combines declarative single agent specification with declarative concurrency between agents. This allows for truly concurrent, distributed systems specified in a declarative fashion. As far as we can tell, this is the first approach of its kind in the field. From the small

examples we have provided it seems like an promising direction.

One of the main open issues at this stage is studying the formal underpinnings of such a system. How does the Join Calculus semantics work together with Jason's practical reasoning semantics. Intuitively there isn't much friction between the two, since each addresses a different concern. Still, representing beliefs as channels and goals as channels and predicates patterns, might prove problematic. In this case we can always fall back to the representation method used by Jason.

Some other, more distant, issues and open directions are:

- Adding a static type system to the language;
- Related to the previous point: using a statically typed *Knowledge Representation Language* (KRL) and studying ontology modeling by using features of the type system;
- Privacy and security issues—investigate the use of the *Principle of Least Privilege* (POLP) [13];
- Tooling for the environment (e.g. IDE, debugger, logger);
- Assessing the tool's use as a means of implementing *Ambient Intelligence* (AmI) scenarios.

This being said, we once again express our belief that this is a fresh, promising approach that, tackled properly, could produce valuable results for the agent community.

REFERENCES

[1] Costin Badica, Zoran Budimac, Hans-Dieter Burkhard, and Mirjana Ivanovic. Software agents: Languages, tools, platforms. *Comput. Sci. Inf. Syst.*, 8(2):255–298, 2011.

[2] Nick Benton, Luca Cardelli, and Cédric Fournet. Modern concurrency abstractions for c#. *ACM Trans. Program. Lang. Syst.*, 26(5):769–804, September 2004.

[3] Gerard Berry and Gerard Boudol. The chemical abstract machine. In *Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '90, pages 81–94, New York, NY, USA, 1990. ACM.

[4] Rafael H. Bordini, Michael Wooldridge, and Jomi Fred Hübner. *Programming Multi-Agent Systems in AgentSpeak using Jason (Wiley Series in Agent Technology)*. John Wiley & Sons, 2007.

[5] William F. Clocksin and Christopher S. Mellish. *Programming in Prolog (4. ed.)*. Springer, 1994.

[6] Alvaro Fernandez Diaz, Clara Benac Earle, and Lars-Ake Fredlund. Erlang as an implementation platform for bdi languages. In *Proceedings of the eleventh ACM SIGPLAN workshop on Erlang workshop*, Erlang '12, pages 1–10, New York, NY, USA, 2012. ACM.

[7] Cédric Fournet and Georges Gonthier. The join calculus: A language for distributed mobile programming. In Gilles Barthe, Peter Dybjer, Luis Pinto, and João Saraiva, editors, *APPSEM*, volume 2395 of *Lecture Notes in Computer Science*, pages 268–332. Springer, 2000.

[8] KoenV. Hindriks. Programming rational agents in goal. In Amal El Fallah Seghrouchni, Jürgen Dix, Mehdi Dastani, and Rafael H. Bordini, editors, *Multi-Agent Programming:*, pages 119–157. Springer US, 2009.

[9] Edmund S.L. Lam and Martin Sulzmann. Finally, a comparison between Constraint Handling Rules and join-calculus. pages 51–66.

[10] Louis Mandel and Luc Maranget. Programming in JoCaml — Extended Version. Research Report RR-6261, INRIA, 2007.

[11] Radek Píbil, Peter Novák, Cyril Brom, and Jakub Gemrot. Notes on pragmatic agent-programming with jason. In Louise Dennis, Olivier Boissier, and RafaelH. Bordini, editors, *Programming Multi-Agent Systems*, volume 7217 of *Lecture Notes in Computer Science*, pages 58–73. Springer Berlin Heidelberg, 2012.

[12] Alessandro Ricci and Andrea Santi. From actors to agent-oriented programming abstractions in simpal. In *Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity*, SPLASH '12, pages 73–74, New York, NY, USA, 2012. ACM.

[13] Fred B. Schneider. Least privilege and more. *IEEE Security and Privacy*, 1(5):55–59, 2003.

[14] Yoav Shoham. Agent-oriented programming. *Artif. Intell.*, 60(1):51–92, March 1993.

Edited by: Dana Petcu
Received: Aug 30, 2013
Accepted: Oct 9, 2013

AIMS AND SCOPE

The area of scalable computing has matured and reached a point where new issues and trends require a professional forum. SCPE will provide this avenue by publishing original refereed papers that address the present as well as the future of parallel and distributed computing. The journal will focus on algorithm development, implementation and execution on real-world parallel architectures, and application of parallel and distributed computing to the solution of real-life problems. Of particular interest are:

Expressiveness:

- high level languages,
- object oriented techniques,
- compiler technology for parallel computing,
- implementation techniques and their efficiency.

System engineering:

- programming environments,
- debugging tools,
- software libraries.

Performance:

- performance measurement: metrics, evaluation, visualization,
- performance improvement: resource allocation and scheduling, I/O, network throughput.

Applications:

- database,
- control systems,
- embedded systems,
- fault tolerance,
- industrial and business,
- real-time,
- scientific computing,
- visualization.

Future:

- limitations of current approaches,
- engineering trends and their consequences,
- novel parallel architectures.

Taking into account the extremely rapid pace of changes in the field SCPE is committed to fast turnaround of papers and a short publication time of accepted papers.

INSTRUCTIONS FOR CONTRIBUTORS

Proposals of Special Issues should be submitted to the editor-in-chief.

The language of the journal is English. SCPE publishes three categories of papers: overview papers, research papers and short communications. Electronic submissions are preferred. Overview papers and short communications should be submitted to the editor-in-chief. Research papers should be submitted to the editor whose research interests match the subject of the paper most closely. The list of editors' research interests can be found at the journal WWW site (`http://www.scpe.org`). Each paper appropriate to the journal will be refereed by a minimum of two referees.

There is no a priori limit on the length of overview papers. Research papers should be limited to approximately 20 pages, while short communications should not exceed 5 pages. A 50–100 word abstract should be included.

Upon acceptance the authors will be asked to transfer copyright of the article to the publisher. The authors will be required to prepare the text in LaTeX 2ε using the journal document class file (based on the SIAM's `siamltex.clo` document class, available at the journal WWW site). Figures must be prepared in encapsulated PostScript and appropriately incorporated into the text. The bibliography should be formatted using the SIAM convention. Detailed instructions for the Authors are available on the SCPE WWW site at `http://www.scpe.org`.

Contributions are accepted for review on the understanding that the same work has not been published and that it is not being considered for publication elsewhere. Technical reports can be submitted. Substantially revised versions of papers published in not easily accessible conference proceedings can also be submitted. The editor-in-chief should be notified at the time of submission and the author is responsible for obtaining the necessary copyright releases for all copyrighted material.