NUI Galway
OÉ Gaillimh

ARAN - Access to Research at NUI Galway

Provided by the author(s) and NUI Galway in accordance with publisher policies. Please cite the published version when available.

Title	SAGE: An Ambient Intelligent Framework for Manufacturing
Author(s)	Hynes, Gearoid; Monaghan, Fergal; Thai, VinhTuan; O'Sullivan, David
Publication Date	2006
Publication Information	Gearoid Hynes, Fergal Monaghan, VinhTuan Thai, Thomas Strang, David O'Sullivan "SAGE: An Ambient Intelligent Framework for Manufacturing", Proceedings of the 9th IFAC Symposium on Automated Systems Based on Human Skill and Knowledge (ASBoHS 2006), 2006.
Item record	http://hdl.handle.net/10379/483

Downloaded 2020-11-26T07:33:21Z

Some rights reserved. For more information, please see the item record link above.

SAGE: An Ambient Intelligent Framework for Manufacturing

Gearoid Hynes, Fergal Monaghan, Vinhtuan Thai, Thomas Strang, David O'Sullivan

Digital Enterprise Research Institute
{firstname.lastname}@deri.org

Abstract: To profit in today's market, small manufacturers must be efficient and agile to minimise cost and to meet variable market demand. This paper discusses an envisaged Semantic, Ambient, Generic and Extensible framework (SAGE) focused on the manufacturing environment. SAGE is based on the ambient intelligence philosophy which means it is focused on the human actor with the aim of aiding them to optimise their companies manufacturing processes. In order to achieve this goal SAGE combines several technologies: semantic reasoning, multi-modal communication and context-aware technologies within a Service Oriented Architecture (SOA). SAGE provides the manager with intuitive access to the real-time data describing the status of the shop floor environment and the ability to automate a response. Two manufacturing processes are targeted for systemic innovation: shop floor control and machine maintenance.
Keywords: Manufacturing, Semantic Web, Context Awareness.

1. INTRODUCTION

Today's market calls for smaller batch sizes of products with a larger range of variants of those products (Heilala and Voho). Market turbulence forces manufacturing plants to constantly adjust their production volume of products, variants and quantities. This is especially true of small to medium sized enterprises (SMEs) that rely on the flexibility and agility provided by their relatively small size to compete with larger competitors.

Deterministic solutions to this problem are intended to solve it in an ideal predictable world, and are inappropriate in a harsh, real-world environment due to the uncertainty of machine failures, the arrival of parts, priority changes and a whole host of changing variables in the manufacturing facility (Ben-Arieh, Chopra et al.). Dynamic solutions involving the gathering of hard real-time data are desirable instead (Ben-Arieh, Chopra et al.). Plant managers must also protect long-term investments in manufacturing systems (Heilala and Voho). Therefore, within an SME, there is a need for flexible and agile manufacturing systems that provide the capability for systemic process innovation to the plant manager. On top of this, individual SMEs may have heterogeneous processes and demands on their manufacturing systems. A system which can be quickly and easily tailored for each SMEs specific process innovation needs is desirable.

The way forward for process innovation in industry lies in the radical innovation of the whole working environment to focus it on the human actor. The Ambient Intelligence (AmI) philosophy sees this human actor surrounded by environments which are sensitive and responsive to their wishes (Aarts). The flexibility and heterogeneity of SMEs demands process innovation systems which have loose-coupling, a generic core, with plug & play extensibility. This paper proposes an AmI approach to provide decision support to the SME user in optimising their manufacturing processes and to more easily introduce new processes. A solution which incorporates technology from the fields of Ambient Intelligence and the Semantic Web to provide systemic innovation to the processes of shop floor control and machine maintenance will be discussed.

The remainder of the paper is organized as follows: Section 2 discusses related work. We present our solution, the Semantic, Ambient, Generic and Extensible (SAGE) framework, in Section 3. Section 4 illustrates the usage of our approach in several scenarios based around three different companies. Section 5 concludes the paper.

2. RELATED WORK

The factory floor application domain has largely been left untouched by the recent wave of ambient intelligence systems research. iShopFloor and

MOSES are two of the few systems for manufacturing that adhere to the Ambient Intelligence philosophy. MOSES (Mobile safety system) (Stottinger 2004) utilizes low-cost passive RFID tags for rotational position sensing. Contextual information is used to enhance the safety of plant maintenance personnel and to provide him with a concise list of tasks to perform in the current location. However, the contextual information derived from this system is limited only to location. Other useful information for maintenance task in manufacturing environment, such as machine state, temperature, etc., is not available.

iShopFloor (Shen, Lang et al.) is an Internet-enabled agent-based intelligent system that provides an open architecture for distributed intelligent manufacturing process planning, scheduling, sensing, and control of the shop floor. The proof-of-concept prototype, however, still lacks an ontology server that can handle inference from semantic data, query answering, and ontology mediating for easy integration of different devices. The Directory Facilitator is a matchmaker between services and resource agents. However, it does not have a central data store to provide information about the status of registered agents. The lack of a central agent registry in this architecture results in communication overhead when active agents try to communicate with inactive agents. The system was also designed for use with Internet browser, which is not always suitable in the shop floor. iShopFloor has limited its potential applications within the shop floor by omitting context awareness capabilities.

There are quite a number of context aware systems primarily focused on the home or office environments (Yan and Selker; Chen, Finin et al.; Tähti, Rautio et al.). The possibility of adapting these existing systems to the manufacturing environment cannot be overlooked. Two of the most prominent context systems are The Context Toolkit and CoBrA. The Context Toolkit (Salber, Dey et al.) is an example of a generic framework which could be adapted for use on the factory floor. It is designed as a basis for developers to integrate context information into their applications without having to worry about the retrieval of that information. This idea did not quite work out as well as had been anticipated because the "widgets" upon which the toolkit was based could not support a wide variety of sensors and hence they were unable to abstract the retrieval away from the developer (Edwards, Bellotti et al.). CoBrA (Context Broker Architecture) (Chen, Finin et al.) is an agent based architecture which uses Semantic Web technologies for sharing information about a context and for reasoning over such information. CoBrA is a tightly coupled system without support for resource discovery of any kind and because of this it is not suitable for adaptation to the SME manufacturing environment as it is too rigid.

3. FRAMEWORK DESCRIPTION

SAGE (Semantic Ambient Generic and Extensible framework) is a framework for the manufacturing environment which combines AmI and Semantic Web technologies to create a highly extensible and loosely coupled framework. In order to achieve this SAGE uses a blackboard model context aware architecture combined with semantically described services.

SAGE is a form of Service Oriented Architecture (SOA). The plug & play extensibility of SAGE depends on the ability to discover and use new services with the minimum of effort. Therefore there must be a standard way of describing what functionality a service brings to the enterprise. Furthermore, this description should be machine-readable to allow for the automatic plug & play extension of SAGE. The automatic and dynamic discovery of services in SAGE is provided by the Web Service Modelling Framework (WSMF) (Fensel and Bussler 2002). WSMF consists of four different main elements for semantically describing Web services: ontologies that provide the terminology used by other elements, goals that define the problems that should be solved by Web services, Web services descriptions that define various aspects of a Web service, and mediation to bypass interpretability problems. SAGE uses the Web Service Modelling Language (WSML) to semantically describe the services. WSMF also has a mechanism for describing the goals that the services may wish to achieve. At the core of WSMF is the Web Service Modelling Ontology (WSMO), which provides a conceptual framework and a formal language for semantically describing all relevant aspects of Web services in order to facilitate the automation of discovering, combining and invoking electronic services over the Web (Roman, Keller et al. 2005). The technical realisation of WSMO, WSMX, is an execution environment that enables discovery, selection, mediation, invocation and interoperation of semantic Web services (Haller, Cimpian et al. 2005).

Fig. 1 shows a high level view of the components of SAGE. At the centre of SAGE are two indispensable components, the Blackboard and the Data Store. Connected to the Blackboard is any number of services with any number of functions. This section describes in detail the Blackboard and the Data Store along with the primary services being used for the manufacturing application.

3.1 Blackboard and Data Store

The key to the extensibility of SAGE is the Blackboard. The functions which the Blackboard performs are relatively straightforward, generic functions that are related to the data in the Data Store, none of which are specific to a particular service. Any specialised functionality is provided by the services that communicate with the Blackboard. The purpose of this is to keep the Blackboard as independent as possible from the application SAGE

is being used for at that particular time. This functional simplicity allows a limitless variety of services to interact with it and hence be included as part of SAGE. If new functionality is required for a specific implementation, specialised services along with any additional ontologies can easily be added to SAGE.

The Blackboard is the only element in SAGE that can access the Data Store. Services pass their information for storage to the Blackboard which can check the data, add it to the store, and perform any updates on the store as necessary. New services register themselves with the Blackboard (see Fig. 1 & 2). Registration involves supplying a semantic description of themselves in the form of WSML to the Blackboard and also registering as listeners to relevant types or subsets of data in the Data Store. Then when there is a change to the data of interest, the Blackboard notifies the listening service, which can take action. The WSML description makes the discovery of services possible to other services, and also allows services to register as listeners to data from stated types of other services.

As triples are added to the store, they form an incomplete ontology. The Blackboard decides when to initiate reasoning over the information to infer further facts about the environment from those asserted by the services. A reasoning program such as Racer Pro runs through the ontology and completes it. Immediately after reasoning, the ontology stored in the Data Store is complete and the framework has a full understanding of the environment at that moment. If services wish to be notified of changes to certain triples in the store, they can register themselves as interested listeners on those triples with the Blackboard. When a change occurs, an event is triggered and the Blackboard notifies the services of the change, at which point they can take action. The Blackboard responsibilities:

- Read/write triples from/to the Data Store
- Initiate reasoning over the ontology in the Data Store
- Notify interested services of relevant changes to data in Data Store

The Data Store that the Blackboard is built on is Yet Another RDF Store (YARS). YARS is a lightweight, open-source, Java-based RDF Store with a small footprint that can be embedded into an application and adapted to meet special application needs (Chen, Finin et al. 2003). Its novel indexing structure allows it to outperform several of its peers: it is 4 to 7 times faster than similar Sesame implementations and up to 400 times faster than the Redland database. YARS stores RDF in the form of triples in N3 notation as opposed to RDF/XML notation, and it also stores the origins of the triples (Harth and Decker 2005). This can be used to track from which service a particular triple of interest came from. Consequently this adds a 4th dimension to the querying that can be performed on the RDF Store: it is possible to search through the information by contributing service. Therefore while the information from all services is massed together into the one store to facilitate context generation, it is still possible to trace where that information came from.

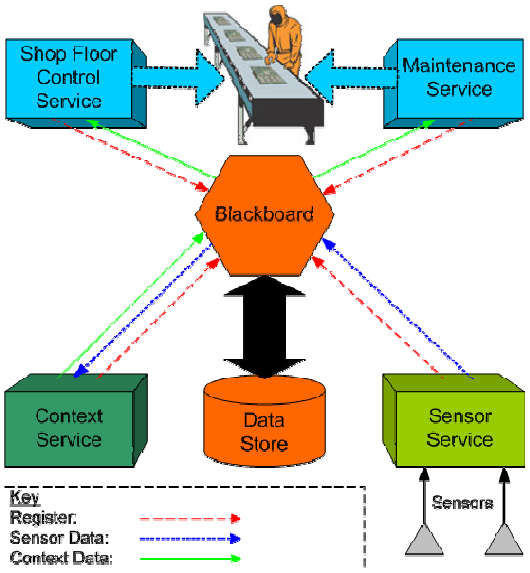

Fig. 1. SAGE Framework generic core components and extensible services

3.2 Sensor Service

The Sensor Service provides the real world knowledge upon which the other SAGE services rely; it is the interface between the framework and physical world. The Sensor Service obtains readings from a series of RFID readers in order to track employees and stock throughout the factory floor. There is an RFID reader located at each workstation and each employee and product batch has a passive RFID tag attached to it. The Sensor Service is responsible for obtaining information from the RFID readers periodically and giving meaning to that data. They do this by translating the readings into meaningful semantic statements in the form of triples. For example, a Sensor Service which is listening to RFID reader on Workstation B can decide to check what tags are in its area, and from the returned value of "52" the Sensor Service construct the triple "Tag52-LocatedAt-Workstation52". The Sensor Service then passes the new triple to the Blackboard for storage in the Data Store and consequent usage by all services.

The Sensor Service does not simply supply data to the Blackboard at pre-determined intervals, it has sufficient intelligence to realise when the sensor reading frequency must be adjusted in order to provide more accurate readings. For instance it can decide that if there is a high variance between subsequent sensor readings to increase the frequency of readings for a short burst to capture what is happening. This can be used to detect irregular activity in the environment or to detect malfunction in other sensors or services that took the readings. In order to achieve this there must be feedback from other services within the framework. The Sensor Service registers itself as a listener with the Blackboard, describing which triples it is interested

in. Then changes to those triples will trigger an event to notify the Sensor Service. Its responsibilities are:

- Take sensor readings periodically
- Translate into triples
- Alter pattern of readings

3.3 Context Service

The purpose of the Context Service is to transform the basic data provided by the Sensor Service into relevant context data. The Context Service is not aware of how the sensor data is placed in the Data Store; it is simply informed by the Blackboard every time new data is available. The Context Service uses both current sensor data and old context data in order to reason the current state of the object in question. Its first step is to take relevant information about the previous context of resources on the shop floor, along with their most recent readings, from the Data Store. Then it calculates a set of statistics about the previous and current states of the resources in question. From these statistics the Context Service can compute the probability of different context states. Based on previously held threshold values for probability, the Context Service then decides the context of the resources. Over time these threshold values can change, providing a capacity to learn to the Context Service.

It is important to note that the Context Service is a step above simply assigning context to resources based on their immediate sensor readings. It has access to information from the Data Store on employees, scheduling, machines and stock etc. that allow it to make truly informed decisions taking into account the past, present and desired future. It can take into account such things as how long an employee must be present at a workstation and why they are at that particular workstation before deciding that that employee is in fact working at that workstation. The ability of the Context Service to learn over time also gives it the ability to perform in a dynamically changing workplace that in practice does not conform to a theoretical or ideal model such as scheduling. Context service responsibilities:

- Calculate statistics
- Compute probabilities
- Decide context
- Learn over time

3.4 Application Services

The purpose of application services is to actuate a response, for the user, to information provided by the other services. The response can be any number of things from presenting data to the user from which they can make decisions to changing the order of production on the shop floor in order to process the orders more efficiently. At present SAGE has two Application Services: Shop-floor Control Service and Machine Maintenance Service.

The Shop-floor Control Service uses information from various sources to control the processing of orders on the factory floor. It primarily relies on information from the Context Service to keep it up to date with the real-time state of the entire factory floor. Using this information and information from the employee roster, order book, stock room and others it controls how orders are routed through the factory and it also controls what workstations employees are working at. The Shop-floor Control Service issues suggestions to the users however they are not obliged to obey them. The framework is sufficiently flexible and intelligent to be able to compensate for a suggestion being ignored. These suggestions are issued via personal digital assistants to the person in charge of scheduling on the factory-floor and if it is necessary to disseminate the suggestions to all the employees it is shown on a series of LCD screens which the factory workers can easily see.

The Shop-floor Control Service also processes the context data in order to provide the user with information on traceability, accountability and reliability:

- Traceability: Orders go through several phases as they progress from when the order is placed to the finished product. In order to maintain 100% traceability it is necessary to know what components went into the product, when and from where those components were supplied, who worked on the product at each stage of its production and who performed the final inspection. The necessary context information is supplied in real-time to the Shop-floor Control Service and it simply stores this information for later retrieval when it is required by the user.
- Efficiency: Using the information stored for traceability the Shop-floor Control Service can calculate various efficiencies. Efficiencies related to employees, stock usage and the assembly line over various periods of time can be provided to the user when required.
- Accountability: As with efficiency, accountability information can be gathered from the traceability information. Full accountability allows the company to know which employee performed which particular task. This allows important information to be gathered such as recognizing individuals with high error rates, most common overall errors, and it allows the company to take measures such as increased employee training to decrease the overall error level and hence increase the overall productivity.

The purpose of the Machine Maintenance Service is to monitor and control all maintenance activities ongoing on the factory floor. It schedules what maintenance is needed where, according to the current maintenance strategy in operation, and when is the most appropriate for this maintenance to occur. It also monitors the effectiveness of the maintenance strategy in operation and it can compare it to other maintenance strategies and if necessary make recommendations to management. As with the Shop-floor Control Service the Machine Maintenance Service issues suggestions via a personal digital assistant which the maintenance team can complete if they wish.

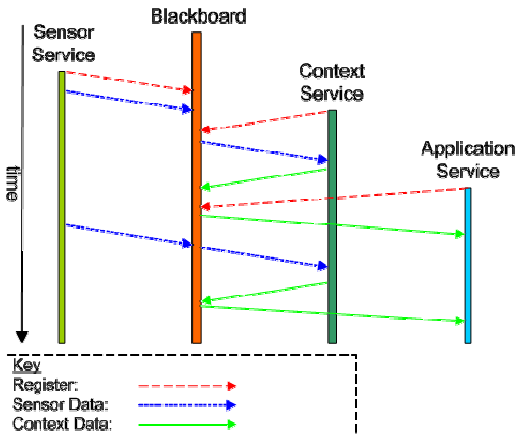

Fig. 2. Interaction of SAGE Services with the Blackboard

Fig 2 gives an example of the interaction between the various services via the Blackboard. Initially the Blackboard is running on its own without any services available. After a period of time the Sensor Service starts-up and registers with the Blackboard, giving it details of the information it will be supplying and the type of information it is interested in. It then supplies the blackboard with SensorData #1 in the form of a triple. Next the Context Service begins; it also registers with the Blackboard in a similar fashion to the Sensor Service. The Blackboard sends SensorData #1 onto the Context Service as it has registered an interest in that data. The Context Service then processes that data and returns ContextData #1 to the Blackboard. An Application Service starts and registers with the Blackboard. ContextData #1 is then forwarded onto it as it is interested in that context data. It then actuates a response in relation to ContextData #1 however this is not reflected on the Blackboard as it is not of concern to the other services. SensorData #2 is supplied by the Sensor Service to the Blackboard who in turn forwards it onto the Context Service. The Context Service uses both SensorData #2 and ContextData #1 to compute the updated context. As before, when the Blackboard receives ContextData #2 it supplies Application Service with it which then actuates a response if necessary.

4. CONCLUSIONS

The capturing and understanding of relevant data from the manufacturing shop floor and the consequent decision-making of a plant manager are very important for small-to-medium sized enterprises to remain agile and flexible in a turbulent market. Deterministic and predictive solutions have drawbacks in that the theories they adhere to in practice do not cope well with the unpredictable nature of market demand and the fast-changing parameters of the factory environment. Our approach overcomes this problem by monitoring the key resources of the shop floor – employees, stock and machinery – in real-time using an ambient intelligent framework. The service oriented architecture can determine context, detect events, actuate responses, and present relevant reports to the user to enable prompt action to be taken. The core of our service framework is generic, modular and extensible to suit the needs of the diverse and ever-changing SME.

REFERENCES

Aarts, E. H., H.; Schuurmans, M. (2001). Ambient Intelligence. The Invisible Future. P. J. Denning. New York, McGraw Hill**:** 235-250.

Ben-Arieh, D., M. Chopra, et al. (1998). Data mining application for real-time distributed shop floor control. IEEE International Conference on Systems, Man, and Cybernetics, San Diego, California.

Chen, H., T. Finin, et al. (2003). An Intelligent Broker for Context-Aware Systems. Adjunct Proceedings of Ubicomp 2003, Seattle, Washington, USA.

Edwards, W. K., V. Bellotti, et al. (2003). Stuck in the Middle: The Challenges of User-Centered Design and Evaluation for Infrastructure. CHI 2003, Ft. Lauderdale, Florida, USA, ACM.

Fensel, D. and C. Bussler (2002). "The Web Service Modeling Framework WSMF." Electronic Commerce: Research and Applications **1**(2): 113-137.

Haller, A., E. Cimpian, et al. (2005). WSMX - A Semantic Service-Oriented Architecture. International Conference on Web Service (ICWS 2005), Orlando, Florida.

Harth, A. and S. Decker (2005). Optimized Index Structures for Querying RDF from the Web. 3rd Latin American Web Congress, Buenos Aires, Argentina.

Heilala, J. and P. Voho (2001). "Modular reconfigurable flexible final assembly systems." Assembly Automation **21**(1): 20 - 30.

Roman, D., U. Keller, et al. (2005). "Web Service Modeling Ontology." Applied Ontology **1**(1): 77 - 106.

Salber, D., A. K. Dey, et al. (1999). The Context Toolkit:Aiding the Development of Context-Enabled Applications. CHI'99, Pittsburgh, ACM Press.

Shen, W., S. Y. T. Lang, et al. (2005). "iShopFloor: An Internet-Enabled Agent-Based Intelligent Shop Floor." IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS, **35**(3): 371-382.

Stottinger, M. (2004). Context-Awareness in industrial environments. Software Engineering. Hagenberg, FH Hagenber**:** 68.

Tähti, M., V.-M. Rautio, et al. (2004). Utilizing ContextAwareness in OfficeType Working Life. MUM 2004, College Park, Maryland USA.

Yan, H. and T. Selker (2000). Context-aware office assistant. 2000 International Conference on Intelligent User Interfaces, New Orleans, LA, January 2000, ACM Press.

.