# Future SCADA challenges and the promising solution: the agent-based SCADA

Hosny A. Abbas

Automatic Control Department,
Qena Paper Company,
Qus, 83621, Qena, Egypt
Email: hosnyabbas@yahoo.com

**Abstract:** SCADA stands for supervisory control and data acquisition; it is a computer system for gathering, analysing and monitoring real time data. SCADA systems are used to monitor and control plants or equipments in a variety of modern industries. SCADA is a critical information system; its criticality comes from the fact that SCADA systems are currently vital components of most nations' critical infrastructures; also failure in SCADA systems may result in catastrophic consequences. SCADA as a critical information system faces the same challenges of current and future information systems such as dynamicity and openness of working environments, efficiency, complexity, and reliability, etc. In this paper, we identify and analyse future SCADA challenges and match it to multi-agent systems as a new software engineering architectural style which will provide SCADA with the abilities and tools to survive in dynamic and open environments and will improve SCADA quality attributes which are the main SCADA challenges. This paper can be considered as a white paper helping readers (SCADA designers and developers) to make a decision about using the agent-based approach for developing future SCADA architectures or continue using traditional software engineering paradigms.

**Keywords:** future SCADA; SCADA challenges; critical infrastructures; critical information systems; agent-based approach; the agent-based SCADA; multi-agent systems and its applications; smart grids.

**Reference** to this paper should be made as follows: Abbas, H.A. (2014) 'Future SCADA challenges and the promising solution: the agent-based SCADA', *Int. J. Critical Infrastructures*, Vol. 10, Nos. 3/4, pp.307–333.

**Biographical notes:** Hosny A. Abbas is currently a PhD student at Faculty of Engineering, Assiut University, Egypt. He received his Master from Aswan Faculty of Engineering, South Valley University, Egypt in 2011, with title 'Efficient web-based SCADA system'. He works for Quena Paper Company in Egypt as an Automation Engineer. His research interests are automation, SCADA, agent-based modelling, multi-agent systems and its applications.

## 1 Introduction

SCADA is an architectural pattern for industrial control systems with many components that are often distributed over a wide area. SCADA is a critical information system; its criticality comes from the fact that SCADA systems are vital components of most nations' critical infrastructures. They control pipelines, water and transportation systems, utilities, refineries, chemical plants, and a wide variety of manufacturing operations. Failure of controlled systems can lead to direct loss of life due to equipment failure or indirect losses due to failure of critical infrastructure controlled by SCADA. SCADA as a critical information system faces the same challenges of other information systems types such as dynamicity and openness of environment, efficiency, complexity handling, robustness, reliability, responsiveness, heterogeneity, security, evolution, flexibility, and safety. SCADA systems are now progressively based on standard information technology utilities and protocols, i.e., TCP/IP, internet, wireless technologies…etc., the reason for that is to achieve interoperability and reduce cost.

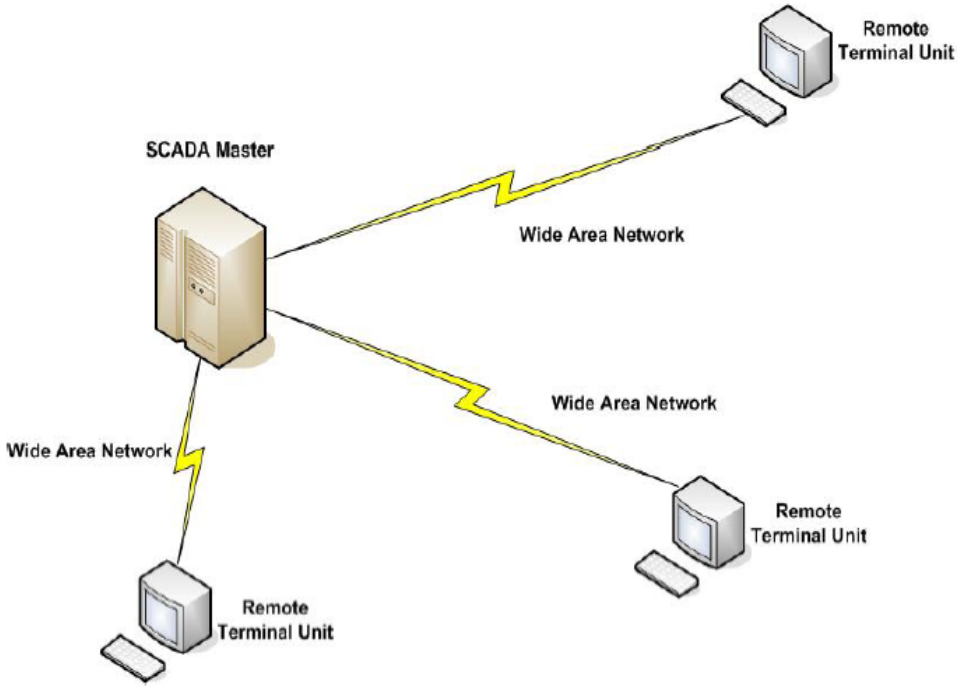
SCADA provides management with real-time data on production operations; implements more efficient control paradigms; improves plant and personnel safety, and reduces costs of operation. These benefits are made possible by the use of standard hardware and software in SCADA systems combined with improved communication protocols and increased connectivity to outside networks, including the internet. SCADA systems use mostly open loop control due to the less reliable communication they use (Fan et al., 2005). It enables remote monitoring and control of a variety of industrial devices as diverse as water and gas pumps, track switches and traffic signals. SCADA systems operators and technical engineers now have more data available than they are capable of managing in the time available to them. In order to manage this amount of data, and allow utility engineers and management to make use of it in an appropriate manner, various systems and architectures have been proposed and developed which aim to integrate the data from remote sites and make it available to users. Many of these are based on client-server methodologies and protocols and many are web-based (Abbas and Mohamed, 2011). SCADA systems have evolved in parallel with the growth and sophistication of modern computing technology. At this time, utilities still purchase SCADA systems and are mostly dependent from SCADA vendors to customise it, at the moment of purchase and later. Even if this is the reality, the present state of the art would allow a different concept of SCADA to be already in daily use. There are many challenges facing current SCADA systems which expand to large geographical distances; we will talk about them, in more details, below. Table 1 shows a chronological description of SCADA evolution till now.

Also Figures 1 to 3 [adopted from NCS (2004)] and Figure 4 adopted from [adopted from Mohamed and Abbas (2011)] shows typical SCADA system architectures for monolithic, distributed, networked, and web-based SCADA systems (Abbas and Mohamed, 2011), respectively.

**Table 1** Chronological description of SCADA evolution

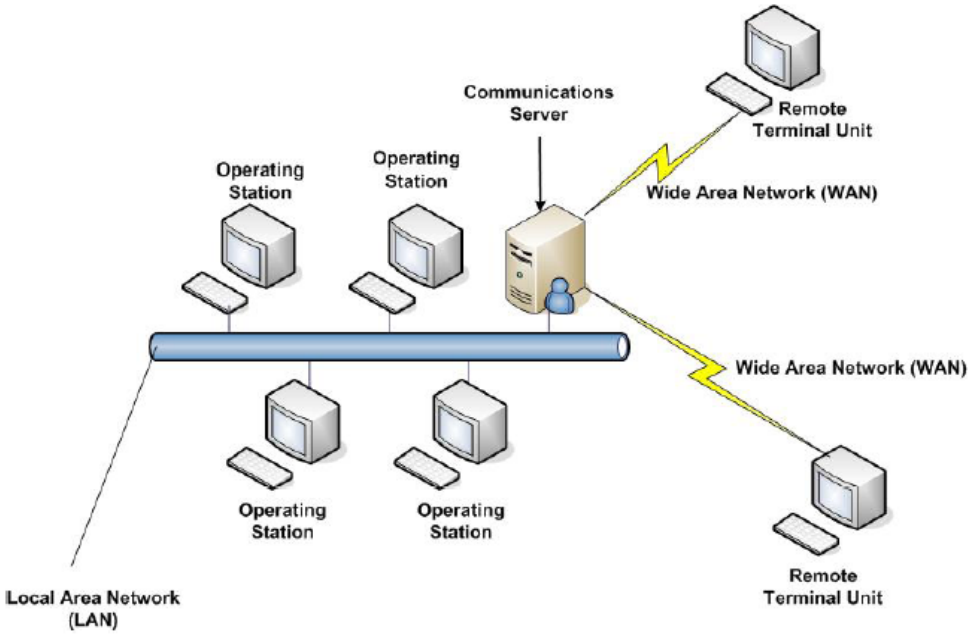
| *Time period* | *Architectural style* | *Description* |
|---|---|---|
| 1970s | Monolithic | Controlled units were on the same site as the controlling computer with hard-wired connections between them (Figure 1). |
| 1980s–1990s | Distributed | SCADA systems networked with devices using special purpose protocols. No external network connection (Figure 2). |
| 2000s | Networked | SCADA systems no longer isolated but connected to external networks, i.e., internet (Figure 3). |
| Currently | Web-based SCADA | Accessing SCADA components from every where at any time using any web browsers, thin clients, PDA, mobile phone, etc. (Figure 4). |
| Future | Agent-based SCADA | Using agents and multi-agent systems new architectural style to build scalable, reliable, and flexible agents. |

**Figure 1** First generation SCADA architecture (monolithic) (see online version for colours)




*Source:* Adopted from NCS (2004)

**Figure 2** Second generation SCADA architecture (distributed) (see online version for colours)




*Source:* Adopted from NCS (2004)

**Figure 3** Third generation SCADA system (networked) (see online version for colours)

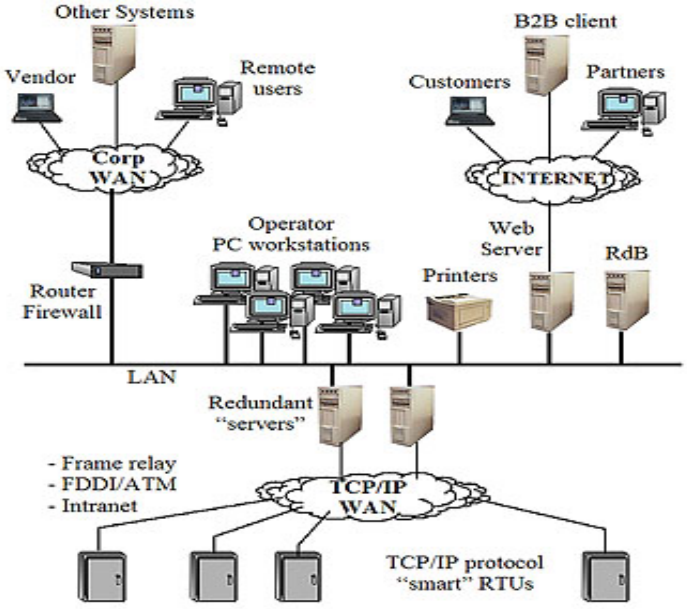
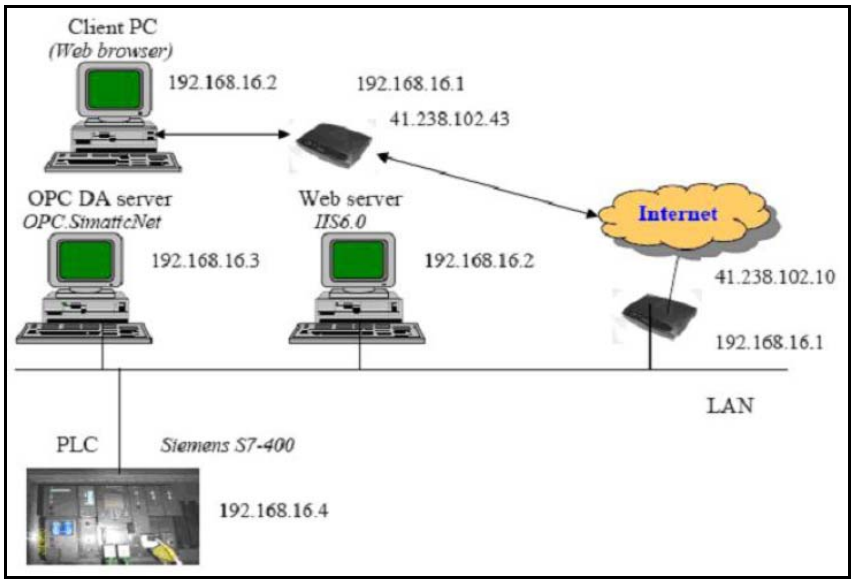



Figure 1.1 – SCADA architecture in the 2000's

*Source:* Adopted from Shaw (2013)

**Figure 4** Currently, web-based SCADA systems (see online version for colours)

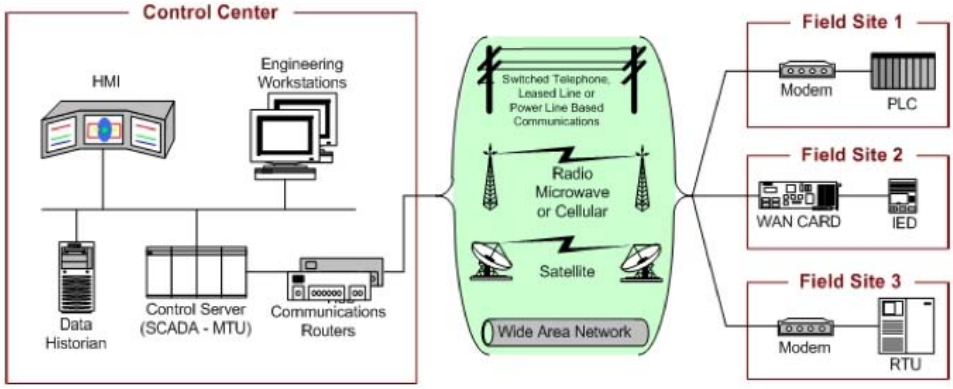



*Source:* Mohamed and Abbas (2011)

Vale et al. (2009) stated that as the required technology exists and diverse factors are urging SCADA systems to a radical change, one can guess that significant changes are about to appear. This means that SCADA is a technology-dependent industrial area; by this we mean that SCADA is a result of the integration of various technology aspects such as computer networks, software engineering, wireless communication, internet and so on. A SCADA control centre performs centralised monitoring and control for field sites over long-distance communications networks, including monitoring alarms and processing status data. Based on information received from remote stations, automated or operator-driven supervisory commands can be pushed to remote station control devices, which are often referred to as field devices. Field devices control local operations such as opening and closing valves and breakers, collecting data from sensor systems, and monitoring the local environment for alarm conditions (Stouffer et al., 2008). Figure 5 shows a practical SCADA architecture used today in many utilities.

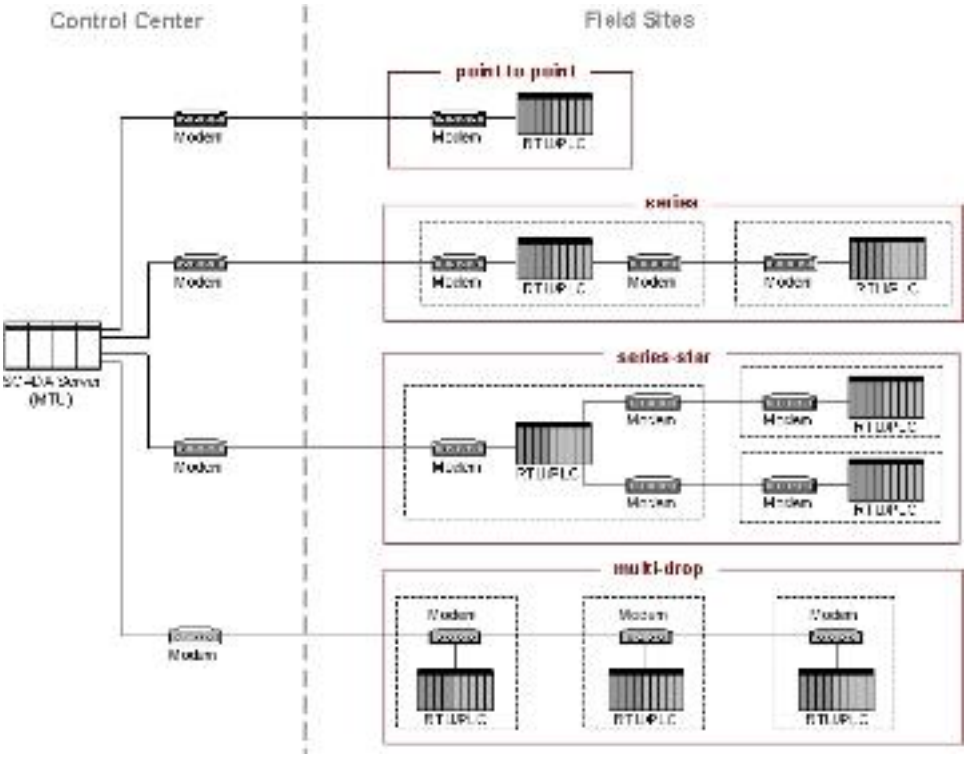
Arghira et al. (2011) claimed that the communication architectures (topologies) are different depending on the implementation. And he described four types of communication architectures currently used: point-to-point, series, series-star, and multi-drop. Point-to-point is functionally the simplest type; however, it is expensive because of the individual channels needed for each connection. Series configuration reduces the number of channels used; however, channel sharing has an impact on the efficiency and complexity of SCADA operations. The series-star and multi-drop configurations use one channel per device which results in decreased efficiency and increased system complexity. Figure 6 shows different types of SCADA communication topologies.

**Figure 5** A practical SCADA architecture (see online version for colours)




*Source:* Adopted from Arghira et al. (2011)

**Figure 6** SCADA communication topologies (see online version for colours)




*Source:* Adopted from Arghira et al. (2011)

An interesting direction of SCADA architectures is what is called web-based or internet-based SCADA systems. Web-based SCADA System makes use of the internet and hypertext transfer protocol (HTTP) and other web technologies as a communication layer of the system. It also uses development tools, framework, platforms and computer languages which are used by regular internet applications as development environment of SCADA application. Web-based SCADA system uses the internet to transfer data between the remote terminal units (RTUs) and the master terminal unit (MTU) and/or between the operators' workstations and the MTU. This will reduce the cost of the installation of the SCADA network if compared with installing a dedicated network for it. It also uses the internet browser programs such as Mozilla Firefox, Netscape Navigator or Microsoft Internet Explorer as graphical user interface (GUI) for the operators HMI. This would give all the benefits of browser-based systems, such as simplifying the installation process of the client side of the SCADA systems and also enable the users to access the system using wide range of platforms, as the browsers is now available in most of the modern operating systems. Abbas and Mohamed (2011) reviewed many web-based SCADA architectures and discussed the pros and cons of each of them.

Chakrabarti et al. (2009) stated that the critical infrastructures, such as electric power systems, telecommunication networks and water distribution networks are systems that influence society's life. Designing, monitoring and controlling such systems is becoming increasingly more challenging as a consequence of the steady growth of their size, complexity, level of uncertainty, unpredictable behaviour, and interactions. Due to the complexity of current industrial distributed systems, such as railways, aerospace systems, navigation systems, gas transmission systems, power utility and power plants, the conventional SCADA system is not capable of providing information management and high-level intelligent approaches. This is because achieving these functionalities requires comprehensive information management support and coordination between system devices, and the control of many different types of task, such as data transportation, data display, data retrieval, information interpretation, control signals and commands, documentation sorting and database searching…etc. These operate at different timescales and are widely distributed over the global system and its subsystems. Without reasonably designed system software architectures and hardware structures, it is impossible to handle these tasks efficiently, safely and reliably, with the possibility of online reconfiguration and flexibly embedding applications (Buse and Wu, 2007). Karnouskos and Colombo (2011) expected that the future infrastructures where a huge amount of data is generated by real world devices and needs to be integrated, processed within a specific context and communicated on demand and on-time, traditional approaches aiming at the efficient data inclusion in enterprise services need to be changed. The main challenges facing current and future SCADA systems related to quality attributes (or non-functional attributes). The extent to which the system possesses a desired combination of quality attributes such as usability, performance, reliability, and security indicates the success of the design and the overall quality of the software system. Software systems quality attributes are the main challenges for developers and designers.

This paper in not intended to provide a real implementation for SCADA systems (we will do that in a second paper) but instead, it is tailored to introduce and analyse the current and future SCADA challenges as a critical information system and substantiates the urgent need of a new architectural style which enables SCADA designers and developers to build large-scale SCADA systems able to survive in dynamic, open and complex working environments. Moreover, we introduce the agent-based approach as a

new software engineering architectural style for developing complex and highly distributed systems. This paper helps SCADA designers and developers to make the decision whether to adopt the agent-based approach or to continue adopting the traditional software engineering paradigms.

This paper is organised as follows; Section 1 gives an introduction to SCADA. Section 2 explains the differences between SCADA, DCS, and PLC; moreover, in this section we propose a conceptual model for automation and SCADA systems. In Section 3 we study and survey current and future SCADA requirements and challenges. In Section 4 we try to match SCADA to multi-agent systems (MASs) as the promising solution to handle SCADA challenges. In Section 5 we describe the attributes of the agent-based approach. Section 6 highlights a modernised electrical grid called 'smart grid' and promotes to use the agent-based approach in architecting them. In Section 7 we present general steps for the development of an agent-based application. In Section 8 we discuss the pros and cons of the agent-based approach. Finally, Section 9 concludes the paper and states the future recommendations.

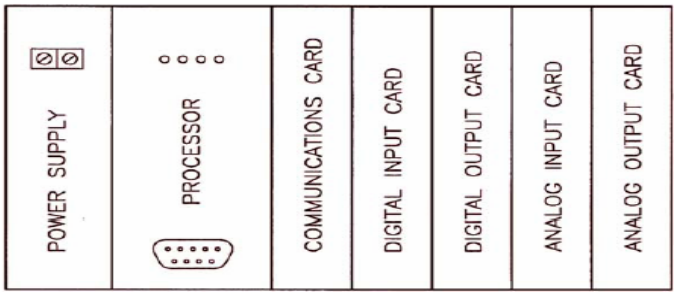
## 2 A conceptual model for automation

Automation is the use of control systems such as numerical control (NC), programmable logic control (PLC), distributed control system (DCS) and other types of control systems, in addition to other applications of information technology such as computer-aided technologies, i.e., CAD, CAM, CAx…etc., to control industrial machinery and processes, reducing the need for human intervention. In the scope of industrialisation, automation is a step beyond mechanisation. Where mechanisation provided human operators with machinery to assist them with the muscular requirements of work, automation greatly reduces the need for human sensory and mental requirements as well. There is confusion among automation beginners and students with the differences between PLC, DCS, and SCADA, so we will give a definition and a description of each of them as a step towards removing this confusion and before we introduce our conceptual model of automation and SCADA.

PLC is a pure control system which can work standalone to control a physical process; it can be used as a RTU in a SCADA system, and usually, it is used for small control processes. Historically, PLCs were usually configured with only a few analogue control loops but for processes require hundreds or thousands of loops, a DCS would instead be used. Figure 7 shows a typical PLC organisation. PLC has some disadvantages such as, it lacks the flexibility for expansion and reconfiguration, the operator interface in PLC systems is also limited, moreover, programming PLC by a higher-level languages and/or capability of implementing advanced control algorithms is also limited.

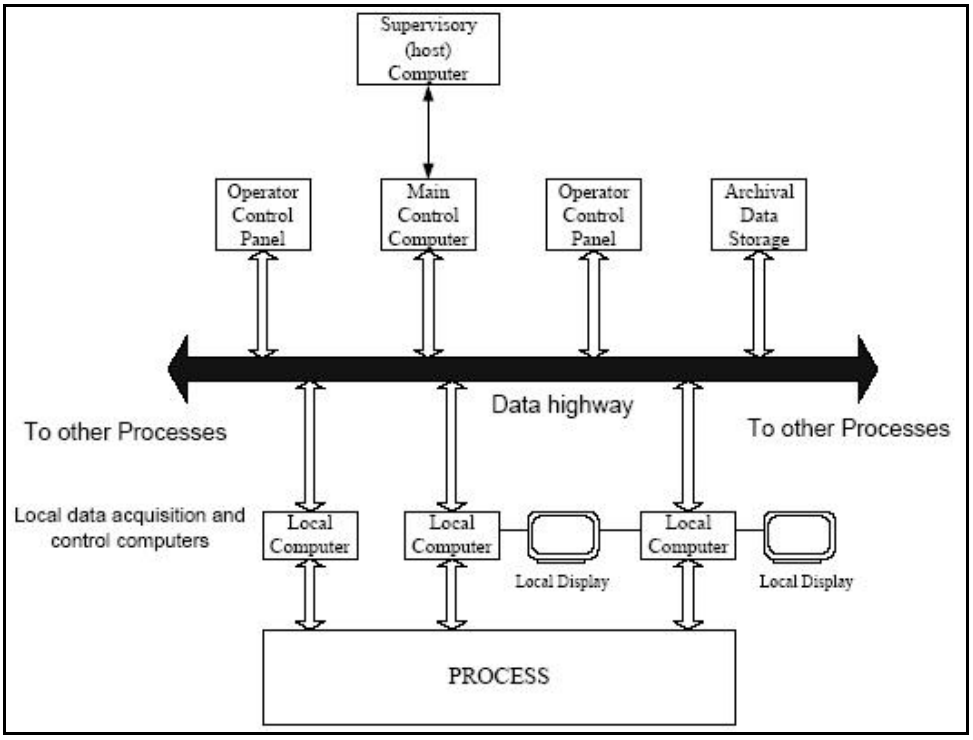
DCS is a system that does what a PLC would do, but the difference is that a DCS is used in much larger and complex control processes. DCS is the system in which controllers are distributed geographically and integrated all the control hardware which is connected from the various field devices. DCS has its own network, Controllers and HMI…etc. It controls the process as a stand-alone system. It has the control loops built into its own controller.

**Figure 7** Typical PLC organisation




The DCS communication path will be through something like a LAN high-speed Ethernet, or other communications network. DCS is a single unit, or a group of local units, it is reliable in the sense that it supports components redundancy; the controller elements of DCS are not central in location but are distributed throughout the system with each component sub-system controlled by one or more controllers. This is a benefit because it prevents failure in one part of the system from affecting another part. Moreover, DCS can start small and expands as needs require. The main disadvantage of DCS is that it is still implemented locally inside a factory floor and not outside the factory boundaries. Actually DCS can be considered as a small-scale SCADA because it has most of the functions provided by SCADA but locally. Figure 8 shows a typical DCS organisation.

**Figure 8** Typical DCS organisation



SCADA systems as defined above expand to large geographical distances; it can even expand beyond countries boundaries using verity of modern communication technologies and information management techniques. Currently, one can say that SCADA may encapsulate DCS and PLC as its remote RTUs. To remove the confusion between SCADA, DCS, PLC, we quote the concept 'automation stack (AS)' and define it as a conceptual model which abstracts the general attributes of SCADA systems, the conceptual model consists of three layers stacked on top of each other (as shown in Figure 9), the bottom layer is the low level field measurements from which we collect all the field data like temperatures, levels, flows, speed, etc. The middle layer includes control systems such as PLC, DCS, and other RTUs, in which the local control of the process takes place. The top layer in the automation stack is real time monitoring and supervisory systems which can be hardware or software and in most of cases its software. Actually, the top layer can be divided into many other layers making hierarchical abstraction models for information management, stations coordination, and so on.

A real SCADA system is considered as a concrete realisation of the automation stack conceptual model. A typical concrete SCADA architecture according to our conceptual model for automation stack is shown in Figure 10.

## 3 SCADA requirements and challenges

In this paper we study SCADA requirements and challenges from the viewpoint of the automation stack top layer (see Figure 9) concerns which overlap SCADA with information management, generation, and dissemination. The main challenges facing current and modern SCADA systems are complexity, scalability, security, reliability, flexibility, Interoperability, robustness, and legacy systems. In what follows we give a brief definition of each of those challenges and its reasons and consequences.

- *Complexity* is the quality of being intricately combined. Complexity tends to be used to characterise something with many parts in intricate arrangement. SCADA systems complexity resulted from adding new components such as computers, operator stations, networks, and other types of resources. Moreover, SCADA complexity comes from the increasing amount of exchanged process data and information, in addition to other interactions between system components. There are two main techniques used for handling complexity, the first is decomposition and integration or breaking down the problem into smaller and smaller pieces until each piece can be solved easily and then putting these small solutions together to form solutions to the bigger problems until the overall solution is achieved. The second is abstraction or ignoring details of a problem that are not relevant to what we are currently doing to make it easier to work with the details that are. Abstraction enables us to solve a problem in simpler, more general terms first and then address specifics of the problem in detail (our proposed conceptual model is an example of abstraction process). The increased size and complexity of today's networked SCADA systems has led to the availability of a large amount of data and information of various types, however, it remains difficult to effectively manage the amount of data produced and to convert this data into knowledge to enable engineers to make use of it. Complexity impacts SCADA systems making them inflexible and cannot easily accommodate

new requirements or changes to control process and monitoring equipment. It is hoped that a new architecture might be able to address this shortcoming.

- *Scalability* is the ability of a system, network, or process, to handle a growing amount of work in a capable manner or its ability to be enlarged to accommodate that growth. A perfectly scalable system is one that has a fixed marginal cost to add additional components. There is a relation between scalability and complexity because increased size means increased complexity. SCADA systems historically distinguish themselves from other industrial control systems (ICS) by being large scale processes that can include multiple sites, and large distances. The new SCADA generation should be scalable, in the sense that new components or client interfaces can be add dynamically without impacting the system performance and without shutting down the system and then run it again.

- *Security* is the degree of resistance to, or protection from, harm. It applies to any vulnerable and valuable asset, such as a person, dwelling, community, nation, or organisation. Establishing or maintaining a sufficient degree of security is the aim of the work, structures, and processes called 'security'. For reasons of efficiency, maintenance, and economics, data acquisition and control platforms have migrated from isolated in-plant networks using proprietary hardware and software to PC-based systems using standard software, network protocols, and the internet. The downside of this transition has been to expose SCADA systems to the same vulnerabilities and threats that plague Windows-based PCs and their associated networks. SCADA systems that tie together decentralised facilities such as power, oil, and gas pipelines and water distribution and waste water collection systems were designed to be open, robust, and easily operated and repaired, but not necessarily secure. The move from proprietary technologies to more standardised and open solutions together with the increased number of connections between SCADA systems, office networks, and the internet has made them more vulnerable to types of network that are relatively common in computer security. 'Security by obscurity' is no longer an option for SCADA security. SCADA systems are a network presence and face significant threats and vulnerabilities. SCADA systems were not initially intended to operate within the enterprise environment. Another issue is the inability within SCADA components to deal with the exposure to viruses, worms, and malware that are commonplace today within the enterprise network.

- *Reliability* is the probability of a component or a system under certain conditions and predefined time, to perform its required task. The main reason for the SCADA failure is the communication network failure. PLCs and PCs have low failure rates compared to communication network. The availability of the communication network should be increased for a more reliable SCADA system. The reliability of the software can be increased at the design stage, instead of development stage because the cost to improve the reliability after design would be higher (Yates, 1990).

- *Flexibility* means that the SCADA system is not a point-to-point communication of fixed path, but a communication that can take place between (among) any random two (or more) points at any time (Haijing et al., 2006). Enhancing information

system flexibility can be achieved with flexible information technology infrastructure and adaptable application systems. The client-server model, used by most current systems, is widely supported and therefore provides a simple means to develop a distributed application. However, it is more suited to centralised applications, in which one server serves a number of clients, or one client controls a number of servers, than true distributed applications, and is lacking in flexibility (Neumann and Zdun, 2000).

- *Interoperability* is a property referring to the ability of diverse systems and organisations to work together (inter-operate). The capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units. Interoperability can be achieved through the use of agreed standards and specifications. A major drawback of typical SCADA systems is their inflexible, static, and often centralised architecture, which largely limits their interoperability with other systems. So, it is very important to enable the interoperability and extensibility of future SCADA system.

- *Robustness* is the ability of a computer system to cope with errors during execution or the ability of an algorithm to continue to operate despite abnormalities in input, calculations...etc. Because we should never fully trust in the reliability of electronic equipment, it is important to make sure that SCADA systems are designed with full redundancy and possibly some additional levels of fault tolerance.

- *Legacy systems* are those that continue to be used despite relatively poor performance and a lack of compatibility with other systems. Often, replacing hardware components is an expensive, unpalatable option for the customer. Proprietary SCADA systems tie the customer to one specific control device manufacturer, creating a difficult negotiating position for the customer during future purchases. The new SCADA generation should keep compatibility with those legacy systems for a period of time until a complete update takes place.

**Figure 9** Automation stack, a conceptual model of automation

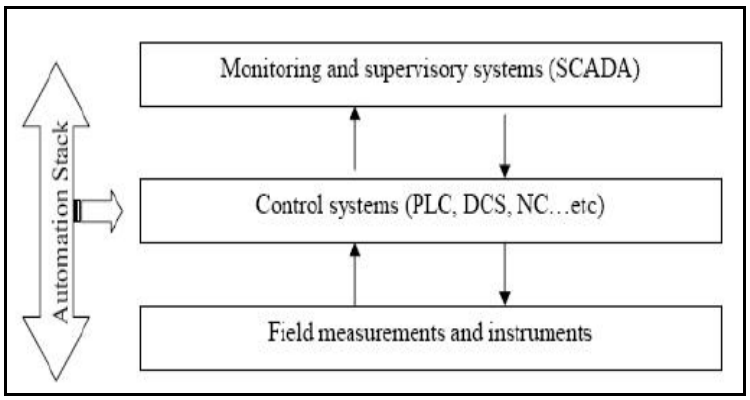
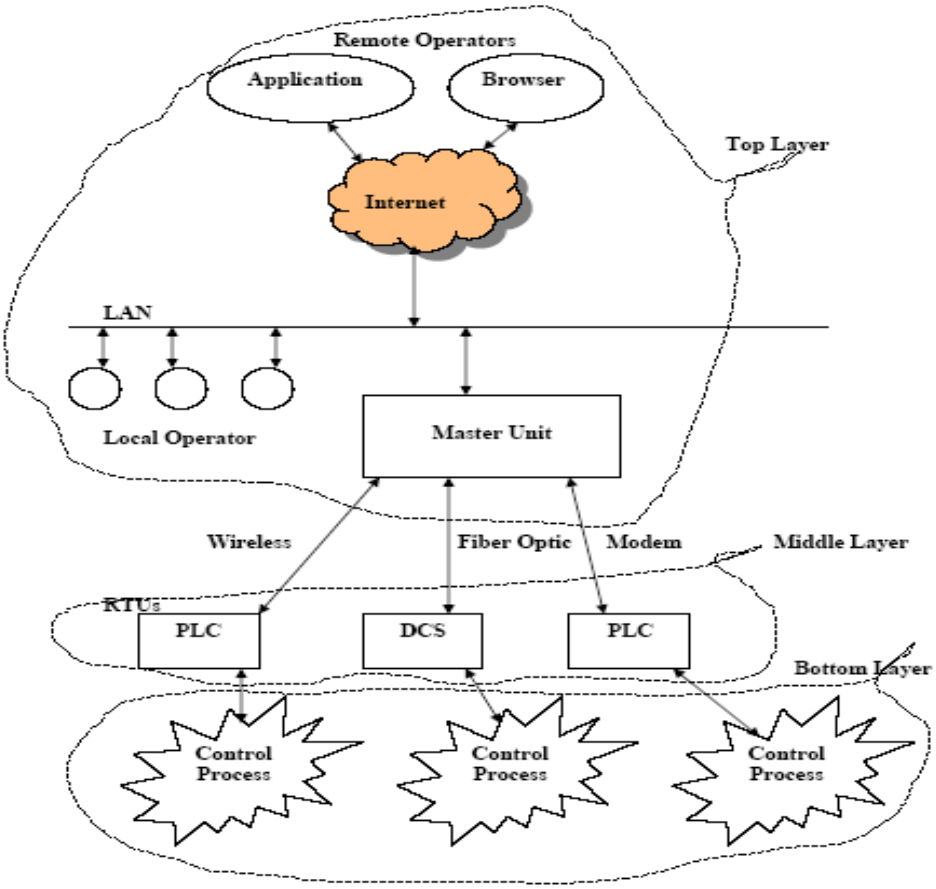


**Figure 10** A typical concrete automation stack (SCADA) (see online version for colours)




## *3.1 Section conclusions*

From the previous discussion about SCADA challenges we conclude that most of them are quality attributes or non-functional requirements that can be handled by using a proper architectural style. It has long been recognised that a system's architecture is critical to the successful design, development, and deployment of large software systems. Architecture has a profound impact on non-functional system properties, such as performance, maintainability, scalability, and dependability. As we mentioned previously that SCADA is an architectural pattern for industrial control systems with many components that are often distributed over a wide area, this means that the system architecture is the key to handle SCADA quality attributes (SCADA challenges). So our goal will be finding a new architectural style which guarantees the design and development of large-scale SCADA systems that are able to adapt with dynamism, openness and uncertainty of working environments such as the internet. Decentralised scheduling will be important to avoid bottlenecks, to keep applications scalable, and to dynamically utilise resources.

MASs as a radically new way of engineering software can be characterised as a new software architecture style. Architectural design is concerned with understanding how a system should be organised and designing the overall structure of that system. In the model of the software development process architectural design is the first stage in the software design process. It is the critical link between design and requirements engineering, as it identifies the main structural components in a system and the relationships between them. Now we will depend on the agents and MASs literature to prove that MASs are the promising solution to handle SCADA challenges and improve its quality attributes, in other words we will try to match SCADA to MASs.

## 4 Matching SCADA to MASs

As we mentioned previously that SCADA is technology-dependent, in the sense that its designers and developers select the suitable contemporary hardware and software technologies to build a flexible, scalable, robust, and secure SCADA systems. Current SCADA faces many challenges which have a bad impact on its performance and efficiency. The reason of this is that current SCADA expanded beyond the boundaries of factories and even countries. Moreover for reasons of efficiency, maintenance, and economics, data acquisition and control platforms have migrated from isolated in-plant networks using proprietary hardware and software to PC-based systems using standard software, network protocols, and the internet. Accordingly, it is expected that the effects of these challenges will increase with future SCADA systems.

Another drawback of current SCADA systems is that they are inflexible and cannot easily accommodate new requirements or changes to control process and monitoring equipments. It is hoped that a new architectural style might be able to address this shortcoming. The client-server model, used by most current systems, is widely supported and therefore provides a simple means to develop a distributed application. However, it is more suited to centralised applications, in which one server serves a number of clients, or one client controls a number of servers, than true distributed applications, and is lacking in flexibility (Neumann and Zdun, 2000). In the future infrastructures where a huge amount of data is generated by real world devices and needs to be integrated processed within a specific context and communicated on demand and on-time, traditional approaches aiming at the efficient data inclusion in enterprise services need to be changed.

Vale et al. (2009) stated that as the required technology exists and diverse factors are urging SCADA systems to a radical change, one can guess that significant changes are about to appear, we claim that the radical change in SCADA architecting and organising can be achieved by the adoption of agents and MASs new software engineering paradigms, according to MASs literature, all SCADA challenges can be solved by this adoption between SCADA and MASs. In the following we will match SCADA challenges to MASs attributes as we found in MASs literature:

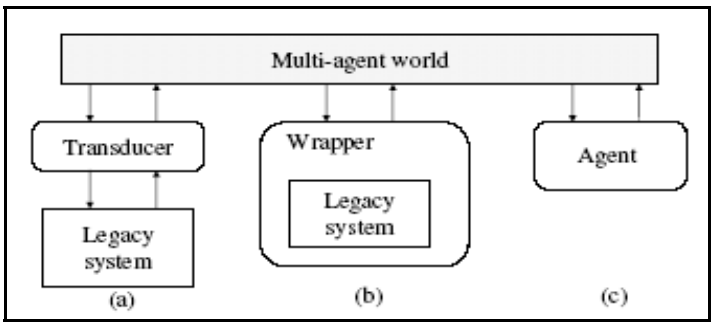
- *Complexity:* MASs are systems of the most representatives among artificial systems dealing with complexity and distribution (Weiß, 1999; Wooldridge, 2002). MASs provide a powerful computational technology, for which dynamic aspects are based on interactions between agents, rather than centralised control. To do their functions the agents have the ability to communicate with others and with the system. Agents

are particularly adapted to complex systems modelling, in which environments are unpredictable and simulation is required. Jennings (2001) stated that "the natural way to a complex system is in terms of multiple autonomous components that can act and interact in flexible ways in order to achieve their set objectives", and also that agents provide a 'suitable abstraction' for modelling systems consisting of many subsystems, components and their relationships.

- *Scalability:* Generally, scalability refers to how well the capacity of a system to do useful work increases as the size of the system increases. Scalability and complexity are in contrast to each other, so succeeding in handling complexity will automatically improve scalability. As MASs are able to deal with complexity, this means that MASs are able to improve systems scalability by using proper MAS architectures and models. MASs also tend to be rapidly self-recovering and failure proof, usually due to the heavy redundancy of components and the self-managed features. They are applied in the real world to many fields such as computer games, information systems and so on, because of their ability to achieve automatic and dynamic load balancing, high scalability, and self-healing systems.
- *Flexibility:* MASs are claimed to be especially suited to the development of software systems that are decentralised, can deal flexibly with dynamic conditions, and are open to system components that come and go. That is why they are used in domains such as manufacturing control, automated vehicles, and e-commerce markets. An agent-based approach provides a flexible, robust and adaptive mechanism for large-scale distributed systems and this is especially helpful when components of the system are not known in advance, change over time, and highly heterogeneous. Jennings (2001) stated that "the natural way to a complex system is in terms of multiple autonomous components that can act and interact in flexible ways in order to achieve their set objectives". Briefly, the main feature which is achieved when developing MASs is flexibility, since a MAS can be added to, modified and reconstructed, without the need for detailed rewriting of the application.
- *Security:* There is a tradeoff between security and performance, traditional security mechanisms such as authentication, encryption…etc., result in performance degradation of an individual machine because of resources constraints. However, agent-based solutions provide inherent availability and security benefits, including the ability to manage systems during a network outage and the ability to manage systems around firewalls with no additional configuration. Rasmusson (1996) used the term hard security for traditional mechanisms like authentication and access control, and soft security for social control mechanisms. Soft security inspired from human social behaviours such as Trust, Reputation …etc. With soft security approaches an agent maintains the credibility assessment of its information sources to assist in the evaluation of incoming information quality. An agent can employ both soft security and hard security to enhance its level of security.
- *Legacy systems:* A legacy system, in the context of computing, refers to outdated computer systems, programming languages or application software that are used instead of available upgraded versions. A legacy system may be problematic, due to compatibility issues, obsoletion or lack of security support. Actually, legacy systems are considered as a challenge to future SCADA systems because many organisations still use them and SCADA developers find difficulties to match their new SCADA

applications to them. Fortunately, the Agent-based approach provides three ways to handle legacy systems as shown in Figure 11. Future agent-based SCADA systems can keep compatibility with those legacy systems for a period of time until a complete updating takes place and rewriting them to be agent-based.

**Figure 11** Agentification methods of legacy systems, (a) transducer (b) wrapper (c) rewrite the system




### *4.1 Section conclusions*

From the above discussion of MASs capabilities and attributes, we claim that the agent-based approach is the promising solution for the development of future SCADA because it has the ability to handle future SCADA challenges such as complexity, scalability, and flexibility…etc. Although those quality attributes are not unique to MASs, combining them in a single system is unique to MASs. This combination results in the suitability of MASs for solving problems where information, location and control are highly distributed, heterogeneous, autonomous (self-controlled) components comprise the system, the environment is open and dynamically changing, and uncertainty is present. Software agents can be configured and tailored for a certain level of specificity in order to reach goals and objectives for configuration audit, security, monitoring, and reporting tasks. MASs can be configured to work collectively with other agents or can be configured to perform specific and individual tasks.

## 5 Agent-based solution

By an agent-based system, we mean the one in which the key abstraction used is that of an agent. We therefore expect an agent-based system to be both designed and implemented in terms of agents. Note that an agent-based system may contain any non-zero number of agents. The multi-agent case where a system is designed and implemented as several interacting agents, is both more general and significantly more complex than the single-agent case. Shehory (1998) stated that in the past few years MASs have emerged, combining research from the field of distributed artificial intelligence (DAI) with a new approach to software engineering. This new paradigm proposes solutions to highly distributed problems in dynamic, open computational

domains. Moreover, Janca (1995) expected that gents are the next major computing paradigm and will be pervasive in every market by the year 2000, and we confirm that this really happened and the agent-based approach is adopted now in variety of areas such as industry, manufacturing, information systems, military projects, spacecrafts, air traffic control, health, education…etc.

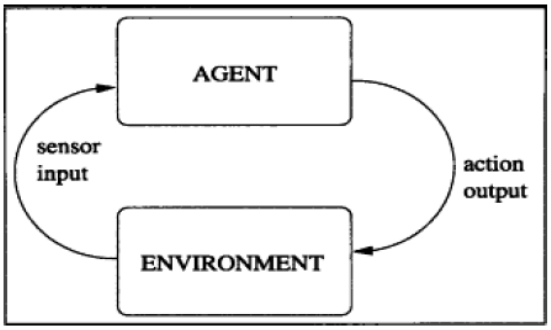
MASs are claimed to be especially suited to the development of software systems that are decentralised, can deal flexibly with dynamic conditions, and are open to system components that come and go. Moreover, they have the ability to achieve automatic and dynamic load balancing, high scalability, and self-healing systems. MASs also referred to as 'self-organised systems'; tend to find the best solution for their problems 'without intervention'. The main feature which is achieved when developing MASs is flexibility, since a MAS can be added to, modified and reconstructed, without the need for detailed rewriting of the application. These systems also tend to be rapidly self-recovering and failure proof, usually due to the heavy redundancy of components and the self-managed features. MASs are one of the most representatives among artificial systems dealing with complexity and distribution. MASs, more than an effective technology, represent indeed a novel general-purpose paradigm for software development. Agent-based computing promotes designing and developing applications in terms of autonomous software entities (agents), situated in an environment, and that can flexibly achieve there goals by interacting with one another in terms of high level protocols and languages. These features are well suited to tackle the complexity of developing software in modern scenarios. Vale et al. (2009) stated that MASs refer to the algorithmic solutions of problems dealing with agents; how agents should interact, avoid conflicts or organise concurrent behaviour (cooperative behaviour in order to fulfil common goals or competitive behaviour). MASs provide a powerful computational technology, for which dynamic aspects are based on interactions between agents, rather than centralised control. To do their functions the agents have the ability to communicate with others and with the system. Agents are particularly adapted to complex systems modelling, in which environments are unpredictable and simulation is required. With MASs as a new software architecture style, it becomes possible to realise software distributed systems have the required properties such as efficiency, complexity handling, robustness, reliability, responsiveness, heterogeneity, security, evolution, flexibility, and safety. Although these properties are not unique to MAS, combining them in a single system is unique to MAS. This combination results in the suitability of MAS for solving problems where information, location and control are highly distributed, heterogeneous, autonomous (self-controlled) components comprise the system, the environment is open and dynamically changing, and uncertainty is present (Shehory, 1998).

Again, the agent-based approach provides a flexible, robust and adaptive mechanism for large-scale distributed systems and that is especially helpful when components of the system are not known in advance, change over time, and highly heterogeneous. Moreover, agent-based methodology also offers the distributed computing solutions. Recently, software agents have become widely used in the modelling of complex, distributed, problems (Jennings, 2001). Each agent is responsible for perceiving the state of environment, updating its own knowledge, deciding future actions and finishing the tasks. There are many definitions of the meaning of agent I found in the literature but the widely accepted definition of an agent is that which stated by Wooldridge and Jennings (1995) who defined agent as:

> "… a hardware or (more usually) software-based computer system that enjoys the following properties: – autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state; – social ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language; – reactivity: agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the Internet, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it; – pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by taking the initiative."

Figure 12 shows a Basic abstract view of an agent (adopted from Weiß, 1999). The Figure shows an agent in its environment. The agent takes sensory input from the environment, and produces as output actions that affect it. The interaction is usually an ongoing, non-terminating one.

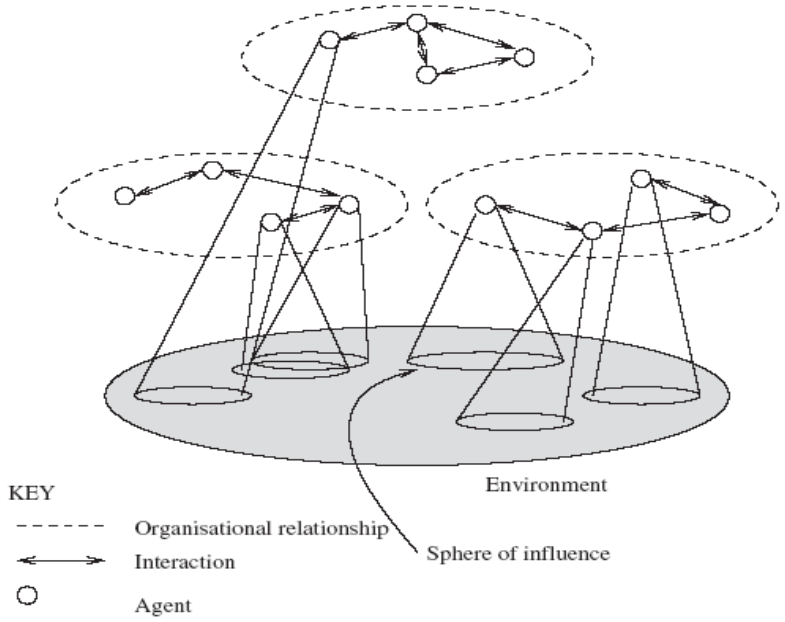
**Figure 12** Basic abstract view of an agent




Source: Adopted from Weiß (1999)

Agent architecture proposes a particular methodology for building an autonomous agent. There are three types of agent architectures, Reactive (reflex) architecture, focused on fast reactions/responses to changes detected in the environment. Deliberative architecture (symbolic), focused on long-term planning of actions, centred on a set of basic goals, Hybrid architecture which combines a reactive side and a deliberative side. A MAS is defined as a set of interacting agents in a common environment in order to solve a common, coherent task. These agents try to achieve individual objectives which are sometimes conflicting (Di Marzo Serugendo et al., 2011). Danny et al. (2006) stated that MASs provide an approach to solve a software problem by decomposing the system into a number of autonomous entities embedded in an environment in order to achieve the functional and quality requirements of the system. A MAS is autonomous, means that there is no external entity which controls this system. This property is enforced because agents inside the system are autonomous. Inside a MAS, data (knowledge) are distributed inside all its agents. Moreover, the control is decentralised (there is no supervisor). Figure 13 shows a typical structure of a MAS adopted from Jennings (2000).

**Figure 13** Typical structure of a MAS




*Source:* Adopted from Jennings (2000)

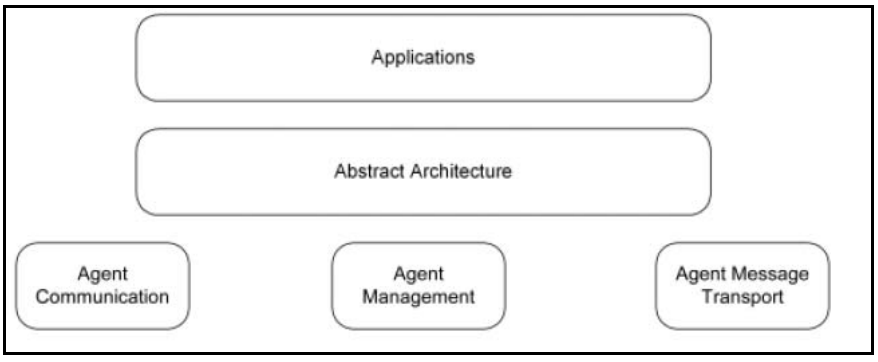
MASs emerged, as a scientific area, from the previous research efforts in DAI started in the early eighties. A MAS is now seen as a major trend in R&D, mainly related to artificial intelligence and distributed computing techniques. This research has attracted attention in many application domains where difficult and inherently distributed problems have to be tackled (Oliveira et al., 1999). The important and promising research topics are, agent modelling, MAS architectures and methodologies, agent communication and interaction engineering, MAS platforms, notation techniques and real world applications. More than this, agent-based computing has been hailed as "the next significant breakthrough in software development" (Sargent, 1992), and "the new revolution in software" (Ovum, 1994). Currently, agents are the focus of intense interest on the part of many sub-fields of computer science and artificial intelligence. Agents are being used in an increasingly wide variety of applications, ranging from comparatively small systems such as e-mail filters to large, open, complex, mission critical systems such as air traffic control. At first sight, it may appear that such extremely different types of system can have little in common. And yet this is not the case: in both, the key abstraction used is that of an agent.

Today, agents are being applied in a wide range of industrial applications. Process control: ARCHON (Jennings et al., 1995), Manufacturing: YAMS (Parunak, 1987) and PABADIS (The PABADIS Consortium, 2002), air traffic control: OASIS (Curtis, 2005) and ADACOR (Leitao and Restivo, 2006), and so on. These applications have been relatively successful, suggesting that the multi-agent approach is a promising method for the implementation of industrial automation systems. However, these systems are intended to focus on a single application or because it is intended for use in manufacturing industries. A new agent-based architecture still needed to provide the

necessary requirements and techniques which are necessary for the development of large-scale critical information systems such as global SCADA systems. SCADA architects should adopt agents and multi-agent software engineering new paradigms to design and develop a new generation SCADA architecture which guarantees to satisfy most of SCADA system current requirements and challenges. The next SCADA generation should be decentralised, flexible, and presents intelligent behaviour (being dynamically adaptive to the context of application domain). The next SCADA generation should have context awareness which is important for achieving intelligent and adaptive SCADA systems. In comparison to client-server and object-oriented systems, MASs have several claimed advantages. Jennings (2001) stated that "the natural way to a complex system is in terms of multiple autonomous components that can act and interact in flexible ways in order to achieve their set objectives", and also that agents provide a 'suitable abstraction' for modelling systems consisting of many subsystems, components and their relationships. Ferber (1999) described how agents, as a form of DAI, are suitable for use in application domains which are themselves widely distributed. The modern SCADA, with remote sites distributed throughout a wide area, falls into this category of systems.

A number of software tools exist that allow a user to implement software systems as agents, and as societies of cooperating agents, by tools we mean agent platforms. An agent platform provides a basis for the implementation of a MAS, and the means to manage agent execution and message passing. For the sake of interoperability, it is intended that the agent platform architecture should be implemented using the Foundation of Intelligent and Physical Agents (FIPA, 2000) specifications of agent platforms abstract architecture. FIPA is an IEEE Computer Society standards organisation that promotes agent-based technology and the interoperability of its standards with other technologies. FIPA, the standards organisation for agents and MASs was officially accepted by the IEEE as its eleventh standards committee on June 8, 2005. The specifications define an abstract agent platform, a number of services that must or may be provided by such a platform and a standard communications language. Figure 14 shows an abstract architecture for FIPA-compliant agent platforms.

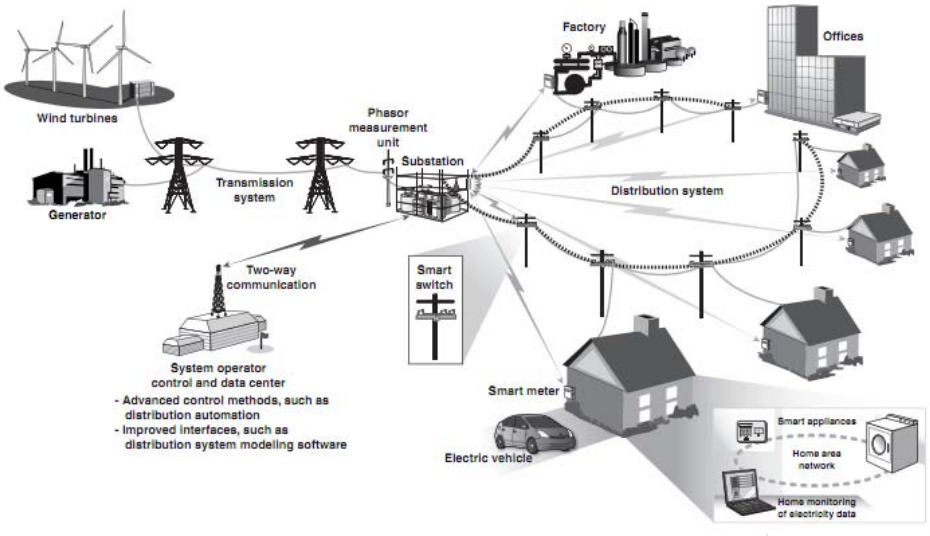
**Figure 14** FIPA abstract architecture for agent platforms



## 6 Smart grids SCADA

A smart grid is a modernised electrical grid that uses information and communications technology to gather and act on information, such as information about the behaviours of suppliers and consumers, in an automated fashion to improve the efficiency, reliability, economics, and sustainability of the production and distribution of electricity (US Department of Energy, 2012). Figure 15 shows an example of smart grid topography.

**Figure 15** The topography of a smart grid




*Source:* Adopted from GAO (2012)

It generally refers to a class of technology people are using to bring utility electricity delivery systems into the 21st century, using computer-based remote control and automation. These systems are made possible by two-way communication technology and computer processing that has been used for decades in other industries. They are beginning to be used on electricity networks, from the power plants and wind farms all the way to the consumers of electricity in homes and businesses. They offer many benefits to utilities and consumers, mostly seen in big improvements in energy efficiency on the electricity grid and in the efficient energy consumption in users' homes and offices. Actually, smart grids are an example of large-scale SCADA systems which will face the same previously mentioned SCADA challenges such as complexity, scalability, reliability, and so on. Although SCADA is a widely used application in most industries, requirements within the electric utility industry for remote control of substations and generation facilities have probably been the driving force for modern SCADA systems (Arghira et al., 2011).

Smart grids are expected to introduce increased system complexity due to multiplicity of interacting entities distributed along large-scale networks under constantly changing environmental conditions and power exchange. The electrical network is composed of a high number of very distributed nodes that are tightly coupled and operating in real time.

Since all the parts of this network have organically grown over many years, even decades, figuring out where intelligence needs to be added is very complex. The Smart Grid represents a technical challenge that goes way beyond the simple addition of an Information Technology infrastructure on top of an electrotechnical infrastructure. Each device that is connected to a Smart Grid is, at the same time, an electrotechnical device and an intelligent node (IEC, 2013). Installing 'smart meters' and upgrading legacy utility networks will force electricity providers to process far more information than they're accustomed to handling.

Here, we repeat what Weiß (1999) and Wooldridge (2002) claimed that MASs are one of the most representatives among artificial systems dealing with complexity and distribution. Moreover, the agent-based approach provides the concept of 'mobile agent' which is a composition of computer software and data which is able to migrate (move) from one computer to another autonomously and continue its execution on the destination computer. Using mobile agents resolves a smart grid from transferring a huge amount of data which represents smart meters measurements in substations and the agent travel to data and makes any computational processing on this data locally this will results in efficient usage of smart grids communications bandwidth. Moreover, a Smart Meter can be designed as an embedded agent to act intelligently, proactively, and autonomously. This promotes the adoption of agents and MASs in designing, developing, and simulating smart grids, what can be called agent-based smart grid or AbSG. Finally, as we expected that agents are the next major computing paradigm and will be pervasive in every market in the near future, smart grid makes no exception on this. Recently there are many researchers have already started working on the development of AbSGs, for instance Jin et al. (2011) and Wijaya et al. (2013). There is more to say about smart grids and its agent-based approach but this is out of the scope of this paper.

## 7 Developing agent-based systems

It became obvious now that new SCADA development approaches should be adopted and these approaches should tackle the SCADA systems non-functional requirements such as reliability, scalability and security…etc. A novel approach for distributed SCADA systems can be obtained by the adoption of the emerging technology, MAS. In case a SCADA developer decided to design and develop an agent-based SCADA system, he will have to follow a certain methodology which supports design and specification of agent systems and provides a clear conceptual framework that enables the complexity of the system to be managed by decomposition and abstraction. The methodology defines a number of phases (steps); each phase (step) results in a model which can be an input to the next phase. For the sake of this paper size, we will not expand in the details of agent-oriented modelling but we suggest the reader to read this reference "the art of agent-oriented modelling" by Sterling and Taveter (2009). A general methodology for developing an agent-based application might contain the following phases:

- *Analysing and identifying the system-to-be requirements (domain model):*
  Requirements analysis includes the formulation of functional requirements of the system as well as eliciting and prioritising of the quality attributes requirements. This is in addition to the identification of all system stakeholders.

- *Design of agent architecture (micro level):* The design of the agent architecture will include a mental agent model and functional agent model. The design of the agent will be according to the required mission of the agent as a SCADA agent and also according to the application domain. The mental model concern the mental or cognitive concepts such goals, plans, beliefs, and intentions, if we decide to use these concepts. On the other hand, the functional model describes the modules from which a single agent is comprised, the relationships between, and the interactions among these modules.
- *Design of Multi-agent architecture (macro level):* In this phase we decide the organisation type of the desired multi-agent SCADA system architecture, i.e., hierarchy, flat organisation, subsumption, or a modular organisation. Hybrids of these and dynamic changes from one organisation style to another are also possible. In addition to organisational issues and combined with them, other MAS properties play a role in their architecture and affect their performance. Among them one can find communication structures and protocols, degree of system openness, level of flexibility, infrastructure services and system robustness (replication).
- *Design of the application domain ontology (data model):* Ontology is a mechanism that describes concepts and the relationships between them. The use of ontology provides a clear and rigorous vocabulary for use in an application domain, which is explicitly defined and separate from the application's implementation to develop Ontology for use in SCADA systems domain, it is necessary to analyse the various concepts and relationships exhibited by components of the SCADA system.
- *Choosing the suitable agent and multi-agent development platform (middleware):* There are a lot of platforms for the development of agent-based application, each one of them has its own philosophy regarding to the agent model and architecture. Most of agent platforms are based on Java language, and some have their own programming language. Actually, we prefer to use a FIPA-compliant agent platform such as Java Agent Development Environment (JADE) for the sake of interoperability. Foundation for Intelligent Physical Agents (FIPA) is an international organisation that is dedicated to promoting the industry of intelligent agents by openly developing specifications supporting inter-operability among agents and agent-based applications.
- *Documentation of the designed agent-based architecture:* Clements et al. (2002) underlined three fundamental uses of architecture documentation; communication among stakeholders, software architecture serves as a basis for system analysis and evaluation, and serves as a means for training.
- *Developing the first prototype of the designed architecture:* Using the previously selected MAS platform we develop a prototype to illustrate the extent to which our agent-based application architecture is successful, and feasible.
- *Evaluating the developed prototype empirically by simulation:* Architectural evaluation is examining software architecture to determine whether it satisfies system requirements, in particular the quality attribute requirements (empirically).

- *Evaluating the developed prototype by a real SCADA system:* The deployment and the real application of the designed, developed, tested agent-based system.

In the agent and multi-agent literature there is an important postulate which had proven to be true, it is "Developing MASs is 95% software engineering and 5% multi-agent systems theory" (Danny, 2010b). Two key concepts recently have a remarkable attention by MASs developers, because they have a dominant influence on the desired system quality requirements (i.e., robustness, scalability, adaptability, reliability…etc). They are *architectural design* and *self-organisation.*

- *Architectural design* is concerned with understanding how a system should be organised and designing the overall structure of that system. In the model of the software development process architectural design is the first stage in the software design process. It is the critical link between design and requirements engineering, as it identifies the main structural components in a system and the relationships between them. The output of the architectural design process is an architectural model that describes how the system is organised as a set of communicating components (Sommerville, 2011; Danny, 2010a). It is known that the plan for implementing functional requirements is detailed in the system design and the plan for implementing non-functional requirements is detailed in the system architecture.
- *Self-organisation* is an attractive way to handle the dynamic requirements in software. It refers to a process where a system changes its internal organisation to adapt to changes in its goals and the environment without explicit external control. Self-organisation often results in emergent behaviour that can be either desirable or undesirable. Due to the dynamism and openness of contemporary agent environments and the ever increasing distribution, complexity and dynamic changes in application requirements, understanding the mechanisms that can be used to model, assess and engineer self-organising behaviour in MAS is an issue of major interest (Di Marzo Serugendo et al., 2005, 2011).

## 8 Pros and cons of the agent-based approach

From the previous discussion we conclude that with MASs as a new software architecture style, it becomes possible to realise software distributed systems have the required positive properties such as efficiency, complexity handling, scalability, robustness, reliability, responsiveness, heterogeneity, security, evolution, flexibility, and safety. Although these properties are not unique to MAS, combining them in a single system is unique to MAS. This combination results in the suitability of MAS for solving problems and build distributed systems where information, location and control are highly distributed, heterogeneous, autonomous (self-controlled) components comprise the system, the environment is open and dynamically changing, and uncertainty is present. There are many benefits of the adoption of agent-based approach such as, distributed parallel processing, reduction in the network load, scalability due to dynamic development, asynchronous operation…etc. But on other hand, Agent-less (traditional) technology are limited in functionality and can only be implemented in one of two ways:

1 using a remote API exposed by the platform or service being monitored

2 directly analysing network packets flowing between service components, and both of them increases system and security risks.

However, there are still some drawbacks of MASs compared to traditional software systems such as limited predictability, the patterns and outcomes of agents' interactions are inherently unpredictable, moreover, predicting the behaviour of the overall system based on its constituent components is extremely difficult, and sometimes impossible, this happens because of the strong possibility of emergent behaviour (Jennings, 2000). Another drawback of agent-based approach is related to reliability, although a MAS is reliable in the sense that there is no single or central point of failure because the processing is distributed among many elements, it is only reliable in a collective sense. The entire system will not fail if a single component in the system fails which means that the individual agent does not matter. In other words, the single agent is certainly not a reliable service provider since it is an autonomous element and has the right to determine its own reaction and responses, and sometimes this is not suitable for computational purposes. In spite of the above mentioned drawbacks of MASs, we claim that it is possible to overcome all of them using a good modelling, architecting and organisation techniques.

## 9 Conclusions

The purpose of this paper is to analyse and describe future SCADA requirements and challenges which promote the need of new SCADA architectures able to survive in open and dynamic working environments. Also, this paper shows that agents and MASs software engineering new paradigms should be adopted by SCADA architects and developers to build future SCADA systems. The multi-agent approach provides increased autonomy by giving each agent its own thread of control, and provides asynchronous message-passing. MASs also provide a high-level communications language with a clearly defined semantics, which is useful in information management and integration. A MAS is one of the most representatives among artificial systems dealing with complexity and distribution. MASs, more than an effective technology, they represent indeed a novel general-purpose paradigm for software development. Therefore, the new SCADA generation should adopt the agent-based approach to design and develop dynamic and self-organised architectures for SCADA systems. The agent-based approach enables the developed applications to survive in open and dynamic environments such as the internet. The agent-based approach can be considered as the promising solution for developing SCADA architectures and as an important step towards the next generation of SCADA systems. We intend to design and develop a self-organised SCADA architecture by adopting the agent-based approach and will publish our work results in a second paper. This paper can be considered as a white paper helping readers to make a decision about using the agent-based approach to build and develop future SCADA architectures or continue using traditional software engineering paradigms. However, from our viewpoint, we claim that the agent-based approach matches future SCADA because of its unique ability to improve its quality attributes and handle its challenges.

## References

Abbas, H.A. and Mohamed, A.M. (2011) 'Review in the design of web based SCADA systems based on OPC DA protocol', *International Journal of Computer Networks*, February, Vol. 2, No. 6, pp.266–277, Malaysia.

Arghira, N. et al. (2011) 'Modern SCADA philosophy in power system operation – a survey', *University' Politehnica' of Bucharest Scientific Bulletin*, *Series C: Electrical Engineering*, Vol. 73, No. 2, pp.153–166.

Buse, D.P. and Wu, Q.H. (2007) *IP Network-based Multi-agent Systems for Industrial Automation: Information Management, Condition Monitoring and Control of Power Systems*, *Springer*.

Chakrabarti, S., Kyriakides, E., Bi, T., Cai, D. and Terzija, V. (2009) 'Measurements get together', *IEEE PEM*, Vol. 7, No. 1.

Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R. and Stafford, J. (2002) *Documenting Software Architectures: Views and Beyond*, Addison-Wesley, Boston, MA.

Curtis, K. (2005) *DNP3 Primer*, Revision A, 20 March 2005, DNP Users Group, Canada [online] http://www.dnp.org/About/UsersGroup.aspx.

Danny, W. (2010a) *Architecture-Based Design of Multi-Agent Systems*, Springer-Verlag, Berlin Heidelberg.

Danny, W. (2010b) *Architecture-Based Design of Multi-Agent Systems*, Springer.

Danny, W. et al (2006) *Multi-agent Systems as Software Architecture, AAMAS '06*, May 8–12, Hakodate, Hokkaido, Japan.

Di Marzo Serugendo, G. et al. (2005) 'Self-organization in multi-agent systems', *The Knowledge Engineering Review*, Vol. 20, No. 2, pp.165–189, Cambridge University Press.

Di Marzo Serugendo, G. et al. (2011) *Self-organizing Software, From Natural to Artificial Adaptation*, Springer.

Fan, R., Cheded, L. and Toker, O. (2005) 'Internet-based SCADA: a new approach using JAVA and XML', *The Journal of Comput. Control Eng.*, October, Vol. 16, No. 5, pp.22–26, IET Digital Library.

Ferber, J. (1999) *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Addison-Wesley, Harlow, England.

Foundation for Intelligent Physical Agents (FIPA) (2000) *FIPA Agent Management Specification* [online] http://www.fipa.org/specs/fipa00023/.

Haijing, Y., Yihan, Y. and Dongying, Z. (2006) 'The structure and application of flexible SCADA', *Power Engineering Society General Meeting*, IEEE.

International Electrotechnical Commission (IEC) (2013) [online] http://www.iec.ch/smartgrid/challenges/.

Janca, P.C. (1995) *Pragmatic Application of Information Agents*, BIS Strategic Report.

Jennings, N.R. (2000) 'On agent-based software engineering', *Journal of Artificial Intelligence*, March, Vol. 117, No. 2, pp.277–296, Elsevier Science Publishers.

Jennings, N.R. (2001) 'An agent-based approach for building complex software systems', *Communications of the ACM*, Vol. 44, No. 4, pp.35–41

Jennings, N.R., Corera, J.M. and Laresgoiti, I. (1995) 'Developing industrial multi-agent systems', in *Proceedings of the First International Conference on Multi-agent Systems (ICMAS-95)*, pp.423–430.

Jin, Z., He, Z. and Liu, Z. (2011) 'Multi-agent-based cloud architecture of smart grid', *The Proceedings of International Conference on Smart Grid and Clean Energy Technologies (ICSGCE 2011)*.

Karnouskos, S. and Colombo, A.W. (2011) 'Architecting the next generation of service-based SCADA/DCS system of systems', *IECON 2011 – 37th Annual Conference on IEEE Industrial Electronics Society*, pp.359–364, 7–10 November.

Leitao, P. and Restivo, F. (2006) 'ADACOR: a holonic architecture for agile and adaptive manufacturing control', *Computers in Industry*, Vol. 57, No. 2, pp.121–130, Elsevier.

Mohamed, A.M. and Abbas, H.A. (2011) 'Efficient web based monitoring and control system', accepted in *Proceedings of the Seventh International Conference on Autonomic and Autonomous Systems, ICAS 2011*, May 22–27, 2011, Venice, Italy.

National Communications System (NCS) (2004) 'Supervisory control and data acquisition (SCADA) systems', *Technical Information Bulletin*, October, Vol. 4.

Neumann, G. and Zdun, U. (2000) 'High-level design and architecture of an HTTP based infrastructure for web applications', *World Wide Web*, Vol. 3, No. 1, pp.13–26.

Oliveira, E., Fischer, K. and Stepankova, O. (1999) *Robotics and Autonomous Systems Journal*, 30 April, Vol. 27, No. 1, pp.91–106(16) Elsevier.

Ovum Report (1994) *Intelligent Agents: The New Revolution in Software*.

Parunak, H.V.D. (1987) 'Manufacturing experience with the contract net', in Huhns, M.N. (Ed.): *Distributed AI*, Morgan Kaufmann.

Rasmusson, J. (1996) 'Simulated social control for secure internet commerce', in Catherine Meadows (Ed.): *Proceedings of the 1996 New Security Paradigms Workshop*, ACM.

Sargent, P. (1992) 'Back to school for a brand new ABC', in *The Guardian*, 12 March, p.28.

Shaw, W.T. (2013) 'SCADA system vulnerabilities to cyber attack', *Article, Electric Energy* [online] http://www.electricenergyonline.com/?page=show_article&article=181.

Shehory, O. (1998) *Architectural Properties of Multi-agent Systems*, Technical Report CMU-RI-TR-98-28, The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Sommerville, I. (2011) *Software Engineering*, 9th ed., Pearson Education, Inc., publishing as Addison-Wesley, Ch. 6.

Sterling, L.S. and Taveter, K. (2009) *The Art of Agent-oriented Modelling*, The MIT Press Cambridge, Massachusetts London, England.

Stouffer, K., Falco, J. and Scarfone, K. (2008) *Guide to Industrial Control Systems (ICS) Security*, Final Public Draft, USA.

The PABADIS Consortium (2002) *Pabadis White Paper* [online] http://www.pabadis.org/downloads/pabadis_white_paper.pdf.

US Department of Energy (2012) *Smart Grid / Department of Energy*, Retrieved 2012-06-18.

US Government Accountability Office (GAO) (2012) *Challenges in Securing the Modernized Electricity Grid*, Report [online] http://www.gao.gov/assets/590/588913.pdf.

Vale, Z.A., Morais, H., Silva, M. and Ramos, C. (2009) 'Towards a future SCADA', *Power & Energy Society General Meeting*, *PES '09*, IEEE.

Weiß, G. (1999) *Multiagent Systems, a Modern Approach to Distributed Artificial Systems*, MIT Press, Cambridge.

Wijaya, T.K., Larson, K. and Aberer, K. (2013) 'Matching demand with supply in the smart grid using agent-based multiunit auction', *Fifth International Conference on Communication Systems and Networks (COMSNETS)*, pp.1, 6, 7–10 January.

Woolbridge, M. and Jennings, N.R. (Eds.) (1995) 'Agent theories, architectures, and languages: a survey', in *Intelligent Agents*, pp.1–22, Springer-Verlag.

Wooldridge, M. (2002) *An Introduction to Multi-Agent Systems*, Wiley, New York.

Yates III, W.D. (1990) *The Application of Reliability Engineering Concepts to Software Development*, Thesis, University of Missouri-Rolla, Rolla: UMR.