Audit, Control and Monitoring Design Patterns (ACMDP) for Autonomous Robust Systems (ARS)

Trad, A. & Trad, C.
TradSoft, 1007 Lausanne, Switzerland, antoine.tradsoft@bluewin.ch, catherine.tradsoft@bluewin.ch

Abstract: This paper proposes the Audit, Control and Monitoring Design Patterns (ACMDP) for building Autonomous and Robust Systems (ARS) such as Mobile Robot Systems (MRS). These patterns are also applicable to other Mission Critical and Complex Systems (MCCS). This paper presents a proposal which will help ARS project managers and engineers design, build and estimate the probability that an ARS will succeed or fail. Furthermore, this proposal offers the possibility to ARS problems with the help of audit, monitoring and controlling components, adjust the project management pathways, and define the problem sources as well as their possible solutions, in order to deliver an ARS or an MRS.

Keywords: ARS, ARS or an MRS, design patterns, packages, components, maintenance, audit, control and monitor.

1. Introduction

This paper proposes an ACMDP for building, auditing, controlling and monitoring ARS or MRS from an Information and Communication Technology (ICT) point of view (Trad, A., Kalpic, D. & Fertalj, K., 2002). These “cross-platform” patterns are also applicable to other complex autonomous systems that need high availability and robustness levels. These systems are autonomous in their logic, processing and energy supply (Krishnamurthy, B. & Rexford, J., 2001). The paper results in an ACMDP “requirements proposal” which should help ARS Engineers (ARSE or MRS Engineers) to estimate the probability that the construction and the maintenance of an ARS (or an MRS) will succeed or fail (Trad, A. & Dessimoz, J.-D., 2004). Furthermore, this proposal offers a possibility to forecast possible ARS (and MRS) problems, launch corrective actions as proactive scripts (panic procedures) (Drake C. & Brown, k., 1995), adjust project management pathways and define problem sources as well as their feasible solutions. This possible solution(s) will be handed over to system developers for correction (Watters, P. & Veeraraghawa 2000).

Although we have many languages tools, standards (Papurt, D., 1995) and methods (Burkhardt, R., 1997) for designing and implementing most of ARS and MRS software components, we still do not have «off the shelf specialized» applicable interactive tools, system design methods and Design Patterns (DP) in the areas of building and controlling (Jones, 1996) complex ARS and MRS, or other MCCS as a whole.

This «requirements» proposal is primarily intended to be of interest to higher-level technical personnel, professors, auditors and senior managers in charge of critical ARS and MRS.

1.1 Paper's structure

The paper is composed of the following sixteen sections: The Automated Manufacturing Environments Method (AMEM); AMEM's Team Design (AMEMTD); The Generic and Infrastructural Design Patterns (GIDP); The Specialized Design Patterns (SPECDP); The Communication Design Patterns (COMMDP); The Data Storage Design Pattern (DSDP); The Vision Recognition Design Pattern (VRDP); The Decision Making Design Pattern (DMDP); The Human Robot Interaction Design Pattern (HRIDP); The FES Design Pattern (FESDP); The Audit, Control and Monitoring Design Patterns (ACMDP); The Underlying Architecture; ACMDPs System’s Architecture; ACMP Maintenance; An MRS Implementation; Conclusion and References.
1.2 Acknowledgments
The author assumes that the reader is familiar with basic ARS (or MRS); ICT infrastructure; object oriented; and development methodologies (and some design patterns basics) techniques. We would like to acknowledge the work of the referenced authors of the publications that were used to compile this article. A special thanks to the “ARS Journal” that enabled us to publish this article. Any remaining mistakes or errors in the text are the responsibility of the authors.

2. The AMEM

A “Method” is a procedure for attaining an objective in accordance with a particular theory. Whereas a “Methodology” is the branch of knowledge that deals with a method and its application in a particular field (Brown L., 1997).

Of course this procedure can be related to the “simple” choice of material for a future screw driver product but it can also concern a much less simple automation procedure for car manufacturing process; that is why the AMEM must deliver a “generic” notation and a set of artifacts to describe various types of procedures (Nance R. & Arthur J., 1988).

2.1 ISRQCC the AMEM predecessor

The Information System Risk and Quality Check Coefficient (ISRQCC) is a step-by-step methodology that can help you to plan and implement an iterative process of Information Systems (IS) auditing within any stage of the development, implementation and maintenance of systems from various problem domains. Most organizations will be challenged to use their IS audit and control results in order to change their business operations, re-engineer their IS, or to re-schedule various tasks of project management plans, which could result in automating tasks that might have been performed manually in the past. Various solutions as a result of IS auditing process could be offered: from implementing new development paradigms and emerging technologies, to solutions that are based on legacy systems as a better balance between costs, benefits and risk. However, in all cases the ISRQCC will enable you to have a clearer idea of what your businesses’ and IT needs are, and will give you a clear list of all benefits of using different technologies for developing an IT infrastructure that supports its strategic business objectives.

This might be of significance when knowing that only 26% of software projects succeed when developing IS. Hence the IS audit activities are becoming a very common intervention to ‘save’ them from failure (Trad, A., 1995). The ISRQCC audit and control method is intended to be of interest to non-technical auditors, audit managers, audit committee members, senior managers in charge of critical computing systems such as robot systems, executives, board members, and even seasoned IS auditors (Trad, A., Kalpic, D. & Trad, C., 2002). This audit method results in a heuristic model which will help information system auditors to estimate the probability that an IS will succeed or fail (Dayton, D., 1999). Furthermore the ISRQCC offers the possibility to forecast IS problems, adjust the project management pathways and define the problem’s source(s) as well as its possible solution(s). Although we have many tools and standards (ISACA-S, 1998) for designing and implementing most of the IS components, until today we still do not have applicable interactive tools, methods or theories in the areas of estimating and auditing of risks, costs, feasibility, viability and hence quality of complex IS. The ISRQCC concentrates on the project feasibility and on finding the set of possible solutions for the current and future problems. This paper presents the use of the ISRQCC audit method to audit the system, and help the project manager and designer in establishing competitive maintenance and stabilization procedures. The ISRQCC was used also to define future evolutionary steps; and to avoid the blind and risky method of “let’s re-develop” the whole system. In contrary, it reused as much as possible of the existing system (Trad A. & Kalpic, D., 1999).

2.2 The AMEM proposal

Increasingly competitive “Automated (Autonomous) Manufacturing Environments” (AME) are the main driving forces for research and improvement in the development of avantgard flexible and efficient methodologies based on IS and specialized in the control and monitoring of ARS and MRS (Trad, A., Perrenoud, A., Gauthey, P-F. & Dessimoz, J.-D., Marcuard, J.-D. & Riedi, M., 2004).

The proposed methodology uses various components which all promote an iterative modeling development process of AMEs, based on a high level of reusability of existing components. Similar to existing methodologies that are applied in these application domains (Lewis, R., 2004). The success of such methodologies and the IS development, implementation and maintenance processes strongly influence the way AME processes are managed and carried out; this consequently forces AMEs to evolve and improve.

Many factors affect the AME business evolution and the corresponding IS (re)engineering processes, these factors are based on the organization’s production infrastructure. In this paper the authors propose a methodology for planning and implementing production infrastructures based on DPs specialized in building ARS and MRS; in order to improve AME development process and to fully support company strategic and business needs. From now on, the “AME Methodology” will be identified as AMEM. The proposed methodology uses different components (as set of DPs) which promote an iterative development
process of AMEs, based on a high level of reusability of existing components. Components (such as the COMMDP, please do refer to section -6- for more information) of this methodology have been successfully applied and results are implemented in various projects. The process of gathering and analyzing an application's requirements, and incorporating them into a design procedure, is a complex one and the industry currently supports many methodologies such as the Unified Modeling Language (UML) that define formal procedures for the improvement of AME integration. Regardless of the methodology that is used to perform the analysis and design, UML can be used to express the results (Thrampoulidis, K., 2004).

By using XMI (XML Metadata Interchange, another OMG standard) the UML model (Magnus H.E. & Penker M., 1998) can be transferred from one tool into a repository, or into another tool for refinement. These are the benefits of standardization (Selie, B. & Rumbaugh, J., 1999)!

The success of such methodologies and the IS development, implementation and maintenance processes strongly influence the way AME processes are managed and carried out; this consequently forces AMEs to evolve. AMEs consist in Industrial-grade Personal Computers (IPC), ARSs, PLCs and/or specialized systems. Many factors affect AME evolution and the corresponding IS engineering processes, and many of these factors are based on the production infrastructure of the organization.

In this section the authors propose a methodology for planning and implementing production infrastructures for ARs and of course MRs; in order to improve the AME’s development process that fully supports the company’s strategic and business needs.

The AME Engineers (AMEE) capture the users’ requirements” using standard UML Use Case diagrams (as shown in “Fig. 1.”), and Rational Rose modeling tool can be used for that (www.rational.com, 2004).

Regardless of the technology, the human factor of the team dynamics plays the major role in the success of the ARS (or MRS) design and development team. This factor influences the success an ARS team in all the design, development and maintenance processes. It establishes a starting point for modelling the effect these factors have on team success (Meier, R., 1997).

The author developed the “Team Quality Check Coefficient” (TDQCC) to help project managers and designers in diagnosing some of the problems endemic to ARS (or ICT in general) teams. It can likewise be used to audit and thus to improve the quality of ARS development and maintenance teams. AMEMTD (previously known as the TDQCC) stands in contrast to the typical non-empirical ad hoc policy for team building and evaluation (Trad, A. & Kalpic, D., 1998).

4. The Generic and Infrastructural Design Patterns (GIDP)

An ARS generic and infrastructural design pattern systematically localizes, names, labels, motivates, and explains a general design that addresses a recurring design problem in the construction of ACMDPs basic design patterns. In general, these design patterns are pure ICT DPs that solve design problems, such as the implementation of the Model View Control (MVC) model. It describes the requirement, domain problem, the solution and when to apply the GIDPs. The GIDP is a general arrangement of objects and classes that solve basic ACMDP problem(s). The GIDP is customized and implemented to build the ACMDP in a particular context. The GIDP is strongly influenced by «Gang of Four» (GoF) design patterns (Gamma, E., Helm, R., Johnson, R. & Vlissides, J., 1995).

The GIDP, are elements of the ACMDP reusable basic components. This article is the first version of a series of articles illustrating the ACMDP (known previously as the ISRQCC). In this implementation the author implemented the Abstract Factory, MVC, Factory Method, Builder patterns and the Singleton pattern.

5. The Specialized ARS DP (SPECDP)

Mature engineering disciplines have handbooks and methods that describe successful solutions to known problems. For instance, automobile designers do not design cars using the laws of physics. Instead, they reuse standard designs with successful track records. The extra few percent of performance available by starting from scratch typically isn't worth the cost.

In the previous section we described the GIDP; and presented it as a generic set of ICT DPs. This section proposes an SPECDP that is a specialized set of DPs. Specialized for building specific modules for ARS (or MRS); this SPECDP is applicable to a large variation of scopes, including very abstract fields, such

---

1 Object Management Group (OMG)
as general robotic entities control or enterprises automation, as well as very concrete domains, in particular autonomous ARS or MRS. An SPECDP is a solution to a ICS construct and maintenance problem in an ARS or MRS (or other context); each SPECDP describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that this solution can be reused. SPECDPs are an attempt to describe successful specialized solutions to common ARS problems (such as communication vision control, communication management and other...).

The long term goal is to develop SPECDP handbooks for ARS Engineers (ARSE) (Adelson, B. & Soloway, E., 1985).

From one point of view, there is nothing new about the ARS SPECDP (or ACMDP in general) since by definition patterns capture experience. It has long been recognized that programmers do not think about programs in terms of programming language elements, but in higher-order abstractions (Soloway, E. & Ehrlich, K., 1984).

What is new is that ARSEs are working to systematically document abstractions other than algorithms and data structures. In general, most ARSEs working on patterns are not concentrating on developing object oriented formalisms for expressing patterns for using them (Papert, D., 1995), though a few of them are. Instead, they are concentrating on documenting the SPECDP (or ACMDPs in general) that ARSE use. Relatively few ARSEs thoroughly understand and consistently apply SPECDPs (or ACMDPs in general) “like” DPs in their daily work (Linn, M. & Clancy, M., 1992).

In addition, advances on theoretical grounds for automated cognition show that this type of approach, managing complex systems in numerous small contexts, featuring well-structured, multiple levels of abstractions, is the most appropriate (Dessimoz, J.-D., 2000) (Dessimoz, J.-D., 2003).

The SPECDP proposes a set of DPs aiming to help ARS designers to build and forecast ARS problems, adjust the project management pathways and locate problem sources as well as to define their possible solutions. Although we have many components and modules for designing and implementing most of the ARS software components (Gamma, E., Helm, R., Johnson, R. & Vlissides, J., 1995), until today we still do not have applicable SPECDPs in the areas of control and monitoring of complex ARS or an MRS, that is why we propose the ACMDP that can be added to the companies standard Design Pattern Catalog (DPC).

SPECDP (or ACMDP in general) is a recurring solution to a specialized ARS (or MRS) problem(s). This proposal is intended to be of interest for non-technical personal in charge of critical ARS systems, such as senior managers and auditors in the case of a company or the ARSE for the case of autonomous mobile robots (Trad, A. & Dessimoz, J.-D., 2004).

6. The Communication Design Patterns (COMMDP)

In general, Ethernet based standard « Transmission Control Protocol/Internet Protocol », which is called « TCP/IP », connects and uses a layered synchronous robust high speed communication protocol between any resources of the network that is connected to it. In this section, we propose the « Personal Computer » (PC) based, « Communication Design Pattern » (COMMDP).

COMMDP is based on the TCP/IP protocol to support Remote communication, Control and Monitoring of « Various Delimiters »; such as ARSs and other automated production units. This COMMDP by means of « TCP/IP » can optimally support, use and synchronize the plant production processes.

The COMMDP consists of modules that act as independent systems or components. The intention is to propose robust and low cost PC based COMMDP solutions in order to effectively manage ARS, MRS or production units in a laboratory (or a plant); and to increase the productivity by linking various resources by means of TCP/IP and Ethernet (Trad, A., Gauthey, P-F., Loersch, M. & Dessimoz, J.-D., 2004).

7. The Data Storage Design Pattern (DSDP)

This section presents a proposal of a DP “specialized” for modeling complex and voluminous « Multi-modal Object Data Sets » (MODS), focusing on the specific case of multimodal Audit, Control and Monitoring (ACM) information; with associated user-defined criteria (known also as primary features) and keys. This DP is also used for; inter-tier data exchange, database manipulation and internal data presentation. The system’s criteria understand the raw data, exceptions, application data, application scenarios, system statistics and logging.

The DSDP is a “high level” proposal of a specialized DP for modeling complex, voluminous and stream-oriented MODS; using XML technologies (Bourret R. 2001). Essentially focusing on the specific case of multimodal application or ACM (XML based) data with associated user-defined criteria (known also as primary features) and keys; that the user can manipulate through a specialized Graphical User Interface (GUI) (Pardi, W. 1999).

The DSDP embeds the complex problem of global application data and the related behavior. The « DSDP Model » is assessed using a large collection of entities of the captured design. For the ICS processing tasks, we look at predicting DSDP Quality Criteria (DQC).

This DSDP includes accessing fast-access multimedia database environments during the storage and modification operations, panic handling and control of business processes. The DSDP
Model is assessed using a large collection of annotated information of the ACM processes (Papurt, D. 1995). For the annotation task, we look at predicting DQC on the manipulated specially formatted information (Fowler, M. & Rice, D., 2003). These are some the basic requirements for the DSDP (Drake C. & Brown, k., 1995) (Trad, A., Gauthey, P.-F., Lüthi, C. & Dessimoz, J.-D., 2004).

8. The Vision Recognition Design Pattern (VRDP)

The VRDP proposes a construct for the support of ARS or MRS navigation, employing artificial intelligence constructs (in this case it is the DMDP, that is presented in the next section). The VRDP manages physical cameras and offers an abstraction layer to the captured image(s). The VRDP is linked to the DMDP (two specialized DPs, which are part of the SPECDP) through the ACMDP; and that insures that the ARS (or the MRS) insure:

- Trajectory following.
- Collisions management.
- Object recognition.
- Obstacle avoidance in indoor environments.

Raw images of a specific size are processed one at a time by an image and position recognition algorithm (DMDP) (Trad, A. 1995). The DMDP output signals control directly to the ARS’s motor control system.

9. The Decision Making Design Pattern (DMDP)

The main purpose of the DMDP (as shown in Fig. 2.) is to provide (i) a reasoning and the (ii) basis for a tool that supports the decision making process in various phases of ARS processing, auditing, monitoring and control. The DMDP provides a quality support for dynamics of ARS control of activities (or tasks) that encompass project management planning issues (e.g. scheduling and rescheduling). Complex relationships among execution activities (and tasks) or actions that can be undertaken in order to solve problems encountered before and during ARS processing, control, maintenance or auditing (which may themselves generate further problems). DMDP exhibits an extensive dynamic nature; it can be tailored to be suitable in various problem domains, i.e. any real world environment. This includes finding the best solution for the current ARS problem(s).

However the user can (a) tackle a single or multiple problems, (b) which could be dealt with by DMDP, (c) solutions are suggested by DMDP (d) which in turn are selected or rejected by the user. All encountered problems and a set of their corresponding solutions are stored and categorised within the “ARS task description database” known also as the “ARSDB”.

Fig. 2. Presents the DMDP flow diagram.

However the DMDP should find the best solution for any detected problem, hence DMDP searches for all possible solutions using the “Beam Search Algorithm”. It is very often the case that a selected solution can solve a range of more common types of problems, but its selection could be based on experienced ARSE and therefore includes some of the “tricks” excerpted from previous experiences. Once the solution search procedure (RSRPROC) is initiated, the DMDP system should be capable in a short period of time to provide a whole set of solutions (if any?). The ARS (or ARSE if the system is offline) will have the possibility of choosing one of these solutions and implementing it (which might include rescheduling the ARS execution scenario, control, maintenance, planning tasks if necessary) (Trad, A. 1995).
10. The Human Robot Interaction Design Pattern (HRIDP)

The most desired properties of the ARS or the MRS system are as follows:

♦ The ARS or the MRS should adapt to numerous variations in its environment and yet reach a pre-assigned goal.
♦ The ARS or the MRS must be capable of low-level reactive behavior as well as higher level cognitive performance (re. cognitics).
♦ The ARS or the MRS should be “user-friendly”, and accept very fast and easily new instructions.
♦ The user can discover the ARS or the MRS capabilities through a Graphical User Interface (GUI). This implies that it has a “user-friendly” graphical user interface as shown in “Fig. 3.”
♦ The ARS or the MRS can detect its failures (by interacting with the DMDP and VRDP), possibly continue with a lower-level of expertise (with the help of an external monitoring tool such as FES, as show in “Fig. 5.”), and inform the ARSE for assistance.

A lot of information can also be acquired visually. The ARS or the MRS VRDP manage vision through embedded cameras. In the market there is a wide range of cameras that can be used; currently there are two major groups of interest for us: 1) USB and 2) HTTP/TCP driven cameras.

11. The FES Design Pattern (FESDP)

Once the ARS or the MRS is aware of the presence of another object in its environment, the monitoring module is responsible for keeping up with the object and all that is known about it. This includes monitoring physical characteristics as well as information inferred from the object. Localization and tracking algorithms allow the ARS or the MRS to keep up with the object’s place in space. “Watching” the object includes tracking visual characteristics of objects. The monitoring module is responsible for scheduling and monitoring the data from the tracking. The monitoring module also monitors object communication and interprets input from the object to determine if there is important information being conveyed (Trad, A. & Kalpic, D., 2004).

11.1 What are factors?
This section presents the FESDP (and other ACMDP patterns) direct and indirect factors that directly influence the success of ARS auditing, development, monitoring, control and reengineering. The FESDP (and other ACMDP patterns) has a pool of factors that play a role of major phenomena and determine the ARS (or MRS) quality status and characteristics. A factor identifies the type of problem, which in turn results in corresponding action(s). The FESDP (and other ACMDP patterns) factors are independent of all the other ACMDPs patterns or views.

11.2 What are views?

[Diagram of system views]

Fig. 3: Human Robot Interaction (HRI) is made easier with graphic, real-time e display and simulation properties, mouse-based and control-key functions; multiple real-world oriented debugging aids, as well as high abstraction, hyper parallel command language.

Fig. 4: The system views.
A view is defined as a set of one or more factors. Hence, if \( f_{ij} \) is a factor and \( V_j \) is a view which selects factor \( f_{ij} \) we could define:

- \( f_{11}, \ldots, f_{1r} \) determines View \( V_1 \)
- \( f_{21}, \ldots, f_{2s} \) determines View \( V_2 \)
- \( \ldots \ldots \ldots \ldots \ldots \ldots \ldots \)
- \( f_{m1}, \ldots, f_{mt} \) determines View \( V_m \)

Sets \( F_i \) and \( F_j \) of factors \( f_{i1}, \ldots, f_{in} \) and factors \( f_{j1}, \ldots, f_{jm} \) are not necessarily disjoint sets.

The creation of different views (as shown in figure 4), i.e. group factors, results in different categories of factors.

### 11.3 Proactive monitoring

Proactive monitoring using the FES is essential because every ARS or the MRS, with very few exceptions, is far too complicated to be managed without some kind of automated assistance. Of course the main requirement is autonomy! In fact, automation is implied in the term proactive monitoring (Trad, A., Kalpic, D. & Fertalj, K. 2002).

Monitoring an ARS or the MRS manually or with ad hoc modules, without the help of FESDP or other software tools, is reactive rather than proactive.

FESDP can monitor multiple ARS or the MRS. The system was built using the GIDP MVC pattern (MVCDP) (Von Zimmermann, P, 1992).

---

Fig. 6. The FESDP’s main packages.

As soon as the ARS or an MRS is put into production, it is time to start to maintain and protect it against all possible types of problems.

The following questions must be asked when integrating the FESDP (as shown in “Fig. 6.”) for an ARS or an MRS:

- What kind of problems thus factors are possible?
♦ Which of these problems, hence factors, can cause serious disruption of the ARS or an MRS?
♦ What information needs to be known about the status of the ARS or an MRS to keep it up and running?
♦ When do the support personnel (auditors) need to know about the problems?
♦ How is the support team notified?
♦ What are the panic procedures (Drake C. & Brown, k., 1995)?

The previous issues should be fully managed by the FESDP.
This section suggests the application of FESDP that can provide useful mechanisms to audit control and monitor and notify the eminence of problem occurrence (Jones, 1996).
Notice that the FESDP can be built on a 3rd party basic monitoring infrastructure, for example IPmonitor (IPmonitor, 2004) or Tivoli (Tivoli, 2004).
With respect to the activities and factors, the following requirements, typical for mobile robots, can be stated for the delivery of the FESDP:

♦ **Collisions:** The FESDP must be capable of foreseeing collisions, and of repositioning the ARS or an MRS when that happens.

♦ **Performance:** The focus is on the “Operational Core” specification with respect to the performance of basic tasks. Each basic task must be performed within a precise time sequence that depends on the type of the task. For instance, providing raw materials is a task that must be performed before the production process begins.

♦ **Integrity checking:** Checking the plans validation and resilience.

♦ **Filtering:** Reducing the amount and complexity of data that one needs to review. Examples include log files, error messages, quota reports, and the output from scheduled jobs.

♦ **Health checking:** Attaching probes to essential services so that one is notified immediately when they become unavailable. Examples of services one needs to monitor in this way include web servers and email servers (Krishnamurthy, B. & Rexford, J., 2001). These checks can be also applied to ARS or an MRS ethernet cameras that have embedded webservers.

♦ **Autonomy:** Insuring the system’s autonomy.

♦ **Connection stickiness:** Maintaining connectivity and performance statistics, through the use of graphical displays of the network being monitored. No service is useful if the system on which it is running cannot be accessed.

♦ **Panic procedures:** Assisting in the detection of faults, using emergency procedures (Drake C. & Brown, k., 1995).

♦ **Factors:** Determining which failure factors are affecting the ARS or an MRS.


♦ **Coordinativity:** Agents must be able to coordinate with other agents in order to achieve either a common purpose or simply their local goals. An autonomous system has to coordinate the actions it deliberately undertakes to achieve its designated objective (e.g., collect a set of objects) with some reactions forced on it by the environment (e.g., avoid an obstacle).

♦ **Adaptability:** Agents must adapt to modifications in their environment. They may allow changes to the component’s communication protocol (Vogel, A., Vasudevan, B., Benjamin, M. & Villalba, T., 1999), dynamic introduction of a new kind of component previously unknown or manipulations of existing agents. Application development for autonomous systems frequently requires experimentation and reconfiguration. Moreover, changes in ARS or an MRS assignments may require regular modification.

Factors are managed through the FES management console; the ARSE selects factors that are important for ARS or an MRS execution, as shown in “Fig. 7”.

![Fig. 7. FES management system.](image-url)

**12. The Audit, Control and Monitoring Design Patterns (ACMDP)**

The ACMDP consists of all the previously introduced design patterns. As already written these patterns are defined into two major groups: 1) The GIDP and the 2) SPECDP. Where the main component is the FESDP!
In fact all the other patterns are “auxiliary helpers” to the FESDP.

To improve system audit, monitoring and control; many ways exist, and the one we advocate here is automation, i.e. the use of artificial systems for reaching goals assigned by humans. Leaving aside aspects of energy, and materials, we concentrate here on information processing (Derrien, Y., 1992).

The ACM of an ARS or an MRS are closely related to its scheduling and execution capacities. Controlling is the task of continuously validating and appropriately enforcing the rules of the targeted strategy.

Scheduling, execution, and monitoring of the infrastructure rely on data from various sources, such as sensors, strategies, plans, and command utilities (Simoneau, P., 1999).

In case of failure they must respond in a timely manner to such unexpected, asynchronously received data. It must be emphasized that control here is strictly limited to the main functional system process (Trad, A. & Swissair, 1997).

Any break in normal processing (downtime), any kind of emergency or failure, is likely to divert the support team’s attention completely from all other systems and services that need to be watched.

The alternative to monitoring is waiting for things to break and then responding as required to the resultant emergencies, and of course having downtimes. When problems occur, the support team feels at ease if being able to focus all their attention on the problem at hand, confident that other aspects of operations are not running on the very edge of usability and are not likely to fail as well (Eriksson, H E. & Magnus P., 1998).

How much to monitor, how frequently, and at what expense, all depend on the size and complexity of the ARS or an MRS, on the degree to which the services it supplies can be considered critical, on the time window within which the ARS or an MRS services must be available, and on the size and skill of ARS or an MRS support staff. Proactive system monitoring, is the search for potential problems and their eventual prevention. With reference to ARS or an MRS, the monitoring must be always proactive (Trad, A. & Swissair, 1996).

Watching for evidence of emergence of potential problems and identifying their source, such as checking available disk space, performance, resilience, the quality of essential processes, probing systems to ensure that they are reachable, fulfillment of business strategy and scenario requirements; these all are examples of problem detections before they disrupt the workflow of an ARS or an MRS. This set of defined factors is known as the ARS or an MRS (failure) Factors Estimation Set (FES, please do refer to section -6- for more information on FES and FESDP).

The ACMDP, can be adapted to any monitoring system. A good place to begin is to identify the possible sources of problems (known as factors), which in the ARS or an MRS monitoring system are known as factors (Trad, A., Kalpic, D. & Fertalj, K. 2002).

13. The Underlying Infrastructural Architecture

Client/Server (CS) computing has created a deep paradigmatic shift in the ICS industry; It’s replacing colossal monolithic mainframe systems with applications split across different computation units; in a client and server environment also known as an n-tier system (Trad, A., 2001).

Stateless business objects in the form of XML strings are a paradigm shift within a paradigm shift; this is a new client server revolution within the client/server revolution. Stateless business objects in XML format break-up the client and server sides of an application into independent components that can interact together and roam across networks, with a unique and flexible interface definition (Trad, A. & Kalpic, D., 2001).

13.1 XML Based Systems

XML is an open, text-based mark-up language that provides structural and semantic information to data. XML, a subset of the popular Standard Generalised Mark-up Language (SGML), has been optimised for the Web. This makes XML a powerful, standards-based complement to HTML that could be as important to the future of information delivery on the Web as HTML was to its beginning.

XML is intended to be used by content creators as well as by programmers. Since XML is text-based, it can be read and worked with easily in relatively non-technical situations, but its ability to organise, describe, and structure data also makes it ideal for use in highly technical applications. XML thus provides common ground for creating structured data and making it available for manipulation and exchange between the different tiers (Pardi, W., 1999).

Today the client/server computing systems are replacing the colossal monolithic mainframe systems with components installed on different computation units (clients and servers), due to that the components interface definitions and dependencies became a major problem (Orfali, R., Harkey, D. & Edwards, J., 1996).

That’s why the stateless business objects formatted in XML strings, brought a solution to this serious problem and made an enhancement to the previous client/server technology standards. Stateless business object in XML format, break-up the client and server sides of an application into independent components that can interact together and roam across networks. This interaction is done through one single interface call, thus the frequent data model changes do not imply component interfaces modifications (Albrecht, K., 2000).

13.2 System’s Modeling

One of the main aims of building such a system is “to design it once and to use the resultant design on all three tiers” (Innovator, 2000). To achieve that goal it is very important to respect the “1:1” rule through all three tiers; this rule notes that UML the design (Burkhardt, R., 1997) documents must be identical and stay unchanged between the different tiers. This was fully achieved for
the data modelling part, whereas for the behaviour modelling (Eriksson, H E. & Magnus P., 1998) it was not possible to achieve that goal. Because two different development environments were used on the 1st tier’s client and the 2nd tier’s server. XML is rapidly establishing itself as the meta-grammar for inter-organisational communication around the Internet. It is becoming increasingly urgent that business analysts, systems analysts, and software developers are able to:

♦ Model the information to be represented in XML (data modelling)
♦ Describe the relationships between the XML and the systems to process it (behavior modeling)

Having done so, they must also be able to rapidly generate the boilerplate code associated with the implementing these processes. At present there is no tool capable of doing this(Booch, G., Christerson, M., Fuchs, M. & Koistinen, J., 1999).

13.3 The Object Oriented to Relational Database mapping issues

The object paradigm is based on software engineering principles such as coupling, cohesion, and encapsulation, whereas the relational paradigm (El Masri & R. Navathe, S., 1994) is based on mathematical principles, particularly those of set theory.

The two different theoretical foundations lead to different strengths and weaknesses. Furthermore, the object paradigm is focused on building applications out of objects that have both data and behavior, whereas the relational paradigm is focused on storing data. The "impedance mismatch" comes into play when you look at the preferred approach to access: with the object paradigm, you traverse objects via their relationships, whereas with the relational paradigm you duplicate data to join the rows in tables. This fundamental difference results in a less-than-ideal combination of the two paradigms, but then, a few hitches are to be expected. One of the secrets of success for mapping objects to relational databases is to understand both paradigms and their differences, and then make intelligent trade-offs based on that knowledge (Ambler, S., 2000).

The Object Identifiers (OIDs) and the Unique Object Identifiers (UOID) are used to identify objects and in relational databases they are used as primary keys (Ambler, S., 2000). The Object Oriented (OO) to Relational DataBase Management System (RDBMS) system mapping layer was built to tackle the following issues:

♦ Name mapping filter, for various reasons such as naming conventions, the 2nd and 3rd tier’s data source systems can have different names to identify the same attributes.
♦ Performance, as the system uses the dynamic SQL motor to access relational databases that can cause major performance degradations. Most relational database systems have configuration parameters that set-up the ‘cache’ in order to enhance this type of performance issue. For example the IBM DB2 has the ‘Paragraph’ parameter for that issue.
♦ Referential integrity and OO integrity, which integrity should be taken as base? (LPC Consulting, 1998)


14. ACMDPs System’s Architecture

Our ARS or an MRS is a distributed system built with real-time agents, and is designed to run under MS Windows operating systems; whereas it can be easily ported to Unix based operating systems (Stapleton, J., 1997).

As it is written mainly in Borland C++ (Kolachina, S., 2000) (Hollingworth, J., Gustavson, P., Swart, B. & Cashman, M., 2002) and C#:NET was used for the ACMDP interface (Telles, M., 2002) (Hoque, R., 2000)(Templeman, J. & Olsen, A., 2002), all the components communicate using the TCP/IP supported network.

ARS or an MRS architecture, also known as Multi-Agent System, specify how to accomplish and integrate planning, monitoring, and control into a system constrained by: (1) sensors and actuators, which cause failures and misperceptions; (2) restricted computational resources, which limit the amount of planning and data processing; and asynchronous events (Trad, A., Kalpic, D. & Fertalj, K. 2002), such as failures or errors, which call for immediate attention; and (3) probably the most important hyper-fast processing (In average, it takes about 100 nanoseconds for an agent to take control and perform an elementary operation) (Dessimoz, J.-D., Gauthey, P.-F. & Roulin, P., 2003).

System architectures describe the software components at a macro level in terms of a manageable number of packages inter-related through data and control dependencies (Land, R. & Crnkovic, I., 1999).

The design of such software architectures has been the focus of considerable research for the past decade, which has resulted in a collection of well understood architectural styles and a methodology for evaluating their effectiveness with respect to particular software requirements (Stapleton, J., 1997).

Examples of software qualities include maintainability, modifiability, portability, etc. ARS or MRS architectures can be considered as libraries of components composed of autonomous and proactive agents that interact and cooperate with each other in order to achieve common or private goals.

In this paper we propose a very basic architecture to design the architecture (Zimmermann, P, 1992) of ARS or MRS components and we do not compare it to conventional architectural solutions. The problem focuses on embedded real-time or hyper fast systems.
The ACMDP must deal with external sensors and actuators and must respond in time constants commensurate with the activities of the system in its environment. In our case it is being developed in tight synchronisation with a new, proprietary programming language, which offers unmatched parallelism options.

An autonomous system typically has to accomplish the following operations: acquiring the input provided by sensors, controlling the motion of wheels and other moving parts, and planning of future trajectories.

In addition, a number of factors complicate the tasks: obstacles may block the ARS or an MRS’s path, sensor inputs may be imperfect, the ARS or an MRS may run out of power, mechanical limitations may restrict the accuracy with which the ARS or an MRS moves itself, the ARS or an MRS may manipulate hazardous materials, unpredictable events may leave little time for responding (Hasemann, J-M., 1996). In the ARS or an MRS’s structure, checks and control mechanisms can be integrated at different abstraction levels ensuring redundancy from different perspectives, such as availability, mirroring or even load-balancing.

Contrary to the conventional architecture, checks and controls are not restricted to adjacent layers. Besides, since the structure must permit the separation of the data and control hierarchies, integrity of these two hierarchies can also be verified independently. The joint venture, through its joint manager, proposes a central message server.

The exception and error-management mechanisms, such as wiretapping or supervising can be supported by the joint manager. This manager guarantees non-fallibility, reliability and completeness by the use of robust panic procedures. The most important (key) architectural requirements in engineering the ACMDP packages are:

- **Predictability**: Agents can have a high degree of autonomy in the way they undertake action and communication in their domains. Then it can become difficult to predict individual characteristics as part of determining the behaviour of the system as a whole. For an ARS or an MRS, all the circumstances of the operations will never be fully predictable. The architecture must provide the framework in which an ARS or an MRS can act even when faced with incomplete or unreliable information, like for example contradictory sensor readings.

- **Fallibility-Tolerance**: The failure of a single agent does not necessarily imply a failure of the whole system. Then the system needs to check the completeness and the accuracy of data, information and transactions. To prevent from system failure, different agents can, for instance, implement replicated capabilities. The architecture must prevent the failure of the ARS or an MRS’s operation and the environment. Local problems, like reduced power supply, dangerous vapours, or unexpectedly opened doors should not necessarily imply the failure of a mission.

In this paper we propose an ACMDP and the possible types of architectures are (Eriksson, H E. & Magnus P., 1998):

- **Conventional Architectures** serve for sample classical solutions. Due to lack of space, we only examine the four major conventional architectures - the layered architecture, the distributed, control loops and task trees that can be implemented in an ARS or an MRS.

- **Layered Architecture** is a seven level architecture, where in its lowest level (level 1) reside the ARS or an MRS control routines (motors, joints ...). Levels 2 and 3 deal with the input from the real world. They perform sensor interpretation (the analysis of the data from a single sensor) and sensor integration (the combined analysis of different sensor inputs). Level 4 is concerned with maintaining the ARS or an MRS’s model of the world. Level 5 manages the navigation of the ARS or an MRS. The next two levels, 6 and 7, schedule and plan the ARS or an MRS’s actions. Dealing with problems and re-planning is also part of level 7 responsibilities.

- **Distributed architecture** based on computing systems with components installed on different computation units (clients and servers) is replacing the once prevailing colossal monolithic mainframe systems. The component interface definitions and dependencies are becoming a major problem. A stateless business object in XML format breaks-up the client and server sides of an application into independent components that can interact and roam across networks. This interaction proceeds through one single interface call, thus the frequent data model changes do not imply component interface modifications (Hoque, R., 2000).

- **Control loop** is a controller component to initiate the ARS or an MRS actions. Since an ARS or an MRS has responsibilities with respect to its operational environment, the controller also monitors the consequences of the ARS or an MRS actions adjusting the future plans based on the feed-back information.

- **Task Trees** architecture is based on hierarchies of tasks. Where parent tasks initiate child tasks.

In addition, R. Brooks’s subsumption architecture is also a good answer for some niche, namely the reactive behaviour of animates in situation.

15. ACMP Maintenance

Concerning maintenance, this section proposes the possible application of a monitoring component, where control and monitoring operations may (and frequently do) result in a re-engineering process. Once the ARS or an MRS is put into production, the maintenance and
reengineering considerations start, and the monitoring component should be periodically used to control it. That is an important factor in the ARS or an MRS lifecycle; it may result in a reengineering process that in most cases is due to serious problems and hence high maintenance costs; this process must be closely controlled by the monitoring process (Trad A. & Kalpic, D., 1999) (Jones, 1996).

In line with information technology trends, software languages are evolving towards new paradigms. Consequently, applications written in the past need to be updated and aligned with the state of the art. On most projects, the IS is built just to work and no extra funds are allocated to improve its status. Such IS can cause serious problems, significant budget loses or even major domain problems. The ACMDP reengineering process evaluates the current ARS status and proposes a set of actions to be taken in order to improve the current system. Its main goal is to avoid complete system rebuild. This process recommends a rather iterative and cautious method of conversion to a liveable system. It is recommended that the ACMDP reengineering process responsible with help of the project designer and a team, define the future ARS technical and domain requirements in order to reengineer the ARS in careful paces. Such work can be considered as ARS redesign. Of course, detailed actions must be also added to such a redesign.

Deriving from their own experience, the authors have made an attempt in this paper to propose a global solution. A further step should be to quantify the risk assessment and in activating of corrective measures to improve the probability for a successful IS implementation. An example is the ACMDP that was previously described.

The use of ACMDPs in the ARS or an MRS decreases maintenance costs; ARS no longer evolve as separate entities but are also integrated with each other. The purpose of integrating ARS can be to increase user-value or to decrease maintenance costs. Different approaches, one of which is software architectural analysis, can be used in the processes of integration planning and design (Trad, A. & Swissair, 1997).

16. An MRS Implementation

Let us consider LOMU and LODUR 2004 (as shown in “Fig. 8.”), two mobile autonomous robots implementations that were built for the Swiss Robotics Cup (Trad, A. 2004). They are capable of localizing, picking and throwing small-sized (9cm) “rugby” balls, playing on a 2x3 m table against other robots (Robotics Cup, 2004) (Mobile Robot LOMU, 2004). They are realized with the best current mechatronic and industrial techniques (Ethernet and Modbus field-bus, Beckhoff. PLC with IEC 1131-ST, Sony microPC with Windows XP-Pro and Borland C++Builder, Beck Integrated PC, FPGA, Maxon servomotors, Galil controller, Baumer Electric ultrasound detectors, Axis-Ethernet color camera, optoelectronic gates, Leclanché high-performance batteries, etc.).

Fig. 8. The ARS or an MRS (re. Hornuss 2003, the predecessor of Lomu).

The MRS Lomu won the second position in the Swiss Cup 2004 (SwissCup, 2004); where the author has stabilized the software and improved its robustness (Dessimoz, J.-D. 2004).

17. Conclusion

The design patterns are really useful in allowing us to increase the complexity level of conceptual thinking. Actual ARSEs and MRSEs are for the moment not used to use such technologies. ARS or an MRS engineers and designers, like all other “Information and Communication Technology” (ICT) system designers; in the near future, will have to rely on “Design Patterns” (similar to the ACMDP) to describe, implement and control the architecture of their future system. They also allow designing architectures with built-in traceability, autonomy, flexibility and robustness. We propose that an ARS or an MRS can be conceived as a group of “independent” components (design patterns) that interact to achieve common functional goals and to keep the system up and running using the ACMDP. The ACMDP consists of a set of design patterns, that were previously introduced; and as already mentioned these patterns are divided into two major groups: 1) The GIDP and the 2) SPECDP. Where the main component is the FESDP and represents its nucleus!

In fact all the other patterns are “auxiliary helpers” to the FESDP. The communication between the components is supported using the sockets based (TCP/IP protocol) COMMDP.

This paper’s proposal is made in the form of an ACMDP requirement description that can also be applied to other
autonomous and complex systems. This paper also includes a set of component requirements that are relevant to this ACMDP. Future research directions include formalizing precisely the interactive and proactive structures that have been identified, as well as the sense in which a particular model is treated as an instance of such a style and pattern. We also propose to relate them to existing patterns and lower-level architectural components involving software components, ports, connectors, interfaces, libraries and configurations. We are still working on contrasting our structures to conventional styles and patterns proposed in the software engineering literature.

18. References


Software Maintenance (ICSM), Amsterdam, Netherlands, 2003.