

# The Automation Analysis Pattern

Mohamed E. Fayad, Huáscar A. Sánchez, and Gaston R. Cangiano

**Abstract**— One of the biggest challenges in the field of Software Engineering is that of the automation of complex systems. At the core of this challenge are the questions of “what”, “where” and “how much” to automate in any given system. The concept of automation is pervasive across all domains. Therefore this paper will attempt to produce an analysis pattern that will be applicable wherever automation is present in a system. To do so, we will build on the “Software Stability Model” (SSM) [1,9] of analysis to identify and isolate the core knowledge of the “automation” concept. A set of scenarios will be presented where our pattern is put into action, showing real-world applicability and reusability.

**Index Terms**—Software stability, Software patterns, CRC-Cards.

## I. INTRODUCTION

Automation is virtually present in every area, from industrial production systems to control and regulation of public services, banking, commerce, aviation, etc. Originally, automation was conceived as a way to alleviate or improve the performance of repetitive tasks, which were previously performed by human beings. In [8] Miriam-Webster Online Dictionary defined automation as an automatically controlled operation of an apparatus, process, or system by mechanical or electronic devices which goal is to minimize or substitute human organs of observation, effort, and decision.

Nowadays, with the advent of sophisticated information technology, some applications are born fully automated, and only need minimum supervision and maintenance. For instance, online e-commerce applications are inherently automated and do not necessarily replace a service previously provided by humans since the web is a completely new medium. Telephone sales representatives are still available as an alternative for most companies. At the other end of the spectrum, we have activities that are always performed by humans, but that we would like in the future to be able to automate as much as possible, for instance, driving. On the one hand, some professionals champion the idea of a fully automated world, where human operators are replaced by safe, reliable and productive automation. On the other hand, it is

argued that automation should be carefully introduced into a system at certain levels and only to certain controlled degrees, depending on the characteristics of the system itself, and the nature of the tasks performed by the operators. Namely, it has been proposed [2] that a framework for analysis and design of automation can be conceptually divided into four areas, corresponding to the four fundamental stages of the Human Information Processor Model of performance [3]. These stages are information *acquisition*, *analysis*, action *selection*, and action *implementation*. This partition corresponds roughly to the Supervisory, Control and Data Acquisition Systems (SCADA) [4], which are the type of control software used predominantly in industrial systems automation.

An efficient abstraction for automation therefore should be extracted and be able to encompass the full range of applications along this spectrum. However, studied solutions for automation are oriented generally to cope with a particular domain-specific spectrum; setting a closed boundary full of limitations for a concept that could be implemented in multiple domains. For that reason, when a specific-domain solution is to be implemented in distinct domains, it creates a redundant usage of knowledge and processes across these automation concept’s implementations. This redundant action may arise either, in a long term, an inaccurate solution when this concept is implemented or, in a short term, serious problems when it’s in the actual implementation stage. Thus, being able to represent this concept as a stable and reusable core to span distinct processes or mechanisms, and/or some cognitive behavior (i.e. decision making, planning), it is a valuable task to be considered.

An efficient abstraction for automation therefore should be extracted and be able to encompass the full range of applications along this spectrum. However, studied solutions for automation are oriented generally to cope with a particular domain-specific spectrum; setting a closed boundary full of limitations for a concept that could be implemented in multiple domains. For that reason, when a specific-domain solution is to be implemented in distinct domains, it creates a redundant usage of knowledge and processes across these automation concept’s implementations. This redundant action may arise either, in a long term, an inaccurate solution when this concept is implemented or, in a short term, serious problems when it’s in the actual implementation stage. Thus, being able to represent this concept as a stable and reusable core to span distinct processes or mechanisms, and/or some cognitive behavior (i.e. decision making, planning), it is a valuable task

---

Mohamed E. Fayad and Huáscar Sánchez are with the Computer Engineering Department, College of Engineering, San José State University, One Washington Square, San José, CA 95192-0180 USA, (e-mail: m.fayad@sjsu.edu).

Gaston R. Cangiano is with the Human Factors and Ergonomics Department, College of Engineering, San José State University, One Washington Square, San José, CA 95192-0180 USA, (email:gaston@gastoncangiano.net)

to be considered.

The objectives of this paper are to extract a core abstraction of the automation concept itself that can be reuse in distinct application domains that share the same knowledge or insight; and assure a stable structure that can endure over time. As Christopher Alexander, known as the father of patterns in architectural design, thought of a "timeless way of building", which is not constrained by the materials or methods applicable. We implement these objectives by using Software Stability concepts introduced in [1,9] as a way to identify and isolate Stable Analysis Patterns. Software Stability concepts will partition the core knowledge of this term into Enduring Business Themes (EBTs), Business Objects (BOs), and Industrial Objects (IOs). These artifacts, due to its stable and *reusable* nature, can represent the basis for a pattern definition. For further information on how to identify patterns using Software Stability Concepts, please refer to [9]. The sections below provide a detailed description of the Automation Analysis Pattern.

The process of extracting a stable analysis pattern could be better thought of as a process of "discovery". Such patterns exist in nature; however, their extraction and identification is a challenging process. Along this paper will detail the Automation Analysis Pattern, characteristics, and structure. For instance, section II details the Automation Analysis Pattern Definition. Section III identifies the stable and reusable abstraction of the automation concept itself. Different areas of applications where the automation analysis pattern may be implemented are illustrated in section IV. We conclude this paper in section V.

## II. PATTERN DEFINITION

This pattern represents the core insight of the automating processes, mechanisms, as a whole, through the utilization of mechanical and/or electronic devices that can be spanned in a wide variety of application domains while keeping in mind that certain human intervention may be present (i.e. monitoring).

### **Problem:**

To create an analysis pattern that will be applied across multiple domains where the automation insight is presented or to be introduced in a system. The solution needs to have room for flexibility with respect to *where* and *what* to automate within a given system; at the same time, the pattern should allow for the possibility of full automation as well as partial automation. Thus the main problem we face is how to achieve this challenge and to truly *discover* a stable form for automation.

### **Context:**

Automation is virtually present in every area, from industrial production systems to control and regulation of public services, banking, commerce, aviation, etc. For instance, in a Car manufacturing division where the use of automated process and equipment is prominent in order to accelerate the delivery of a final product (i.e. car). Automation is a term used in multiple domains, which require in certain time an efficient

automation of a process or mechanism, and/or some cognitive behavior, such as decision making and planning.

Also, The automation concept, within its structure, allows the involvement, in certain degree, of human beings to have access to partial or all levels of the automated process, equipment and/or system, and at the same time have enough flexibility to represent them without human intervention or supervisory control only, such as it is the case, for instance, in very complex information processing systems or hazardous situations. Therefore, the creation of a pattern, which captures the core abstractions or knowledge of the automation concept itself, is a challenging task. Keeping in mind, that its actual extraction and/or identification is a constant and difficult task. In this paper we provide the core abstraction of the automation concept aided by the utilization of Software Stability concepts introduce in [9], providing a stable and reusable core that could be spanned in multiple and heterogeneous domains which requires an automation necessity.

### **Forces:**

Following are the challenges we are trying to resolve:

1. The automation concept spans a wide variety of applications distinct in nature. Therefore, this pattern needs to be general enough in order to provide a stable and reusable core for the automation concept itself, and hence, be able to be shared by multiple applications sharing this same knowledge.
2. Automation can be generalized into two aspects: *control* and *monitoring*. Therefore a pattern for automation analysis should be flexible enough to manifest this within its structure.
3. Automation can take effect at various levels on a continuum: from high automation (complete automation) to low automation (only as assistance without direct action, i.e. decision support). There are many levels in-between which vary the amount of autonomy and responsibility of the automation process. Therefore, the definition of a pattern needs to be abstract enough in order to exhibit these levels of execution within its core knowledge.
4. Automation replaces, augments, or supersedes human activities, but is restricted to control and monitoring only, which are the areas that have been shown to work safely. Planning is left to the human operator. Hence, this pattern needs to provide a high level of flexibility for handling different structures and roles in case human intervention is present or not.
5. Automation can control several processes or mechanism concurrently; independently, there can be one or more control and monitoring logic. Therefore, this pattern, within its structure, needs to embody the distinct types of processes and mechanisms involved in the automation process.
6. Automation could be either distributed across a large number of heterogeneous components and over a wide network or conducted locally by several heterogeneous media. This is indeed the trend we are witnessing as distributed systems such as the Global

Automation Platform (GAP) [4] become available to satisfy the demand for companies to share resources. Consequently, this pattern must provide an abstract definition of the distinct media involve in the automation process.

### III. PATTERN SOLUTION

The following model represents the proposed solution for the automation analysis pattern following the semantics of Software Stability Concept approach. See Figure 1. Figure 2 shows the sequence diagram of this analysis pattern.

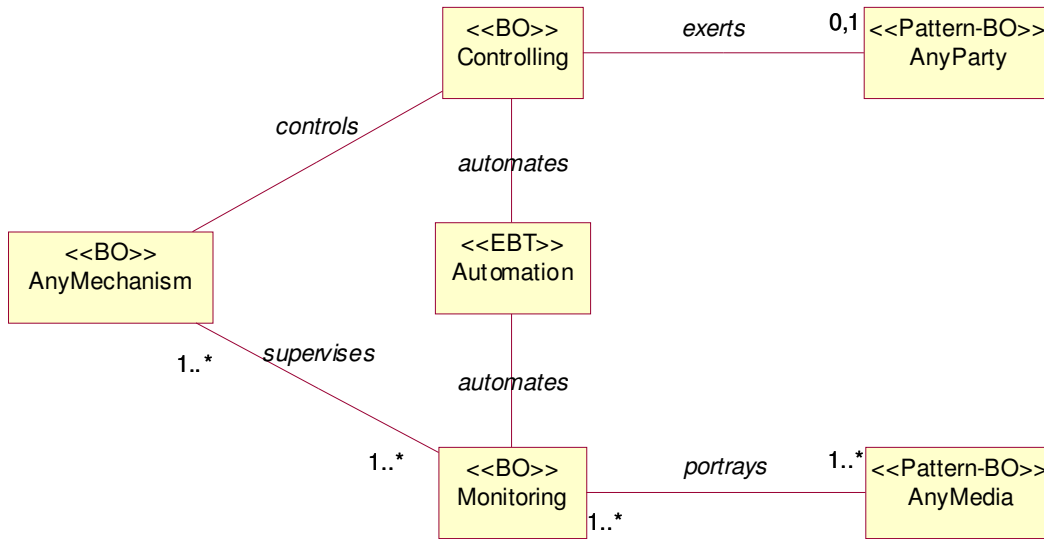


Figure 1: The Automation Analysis Pattern Model

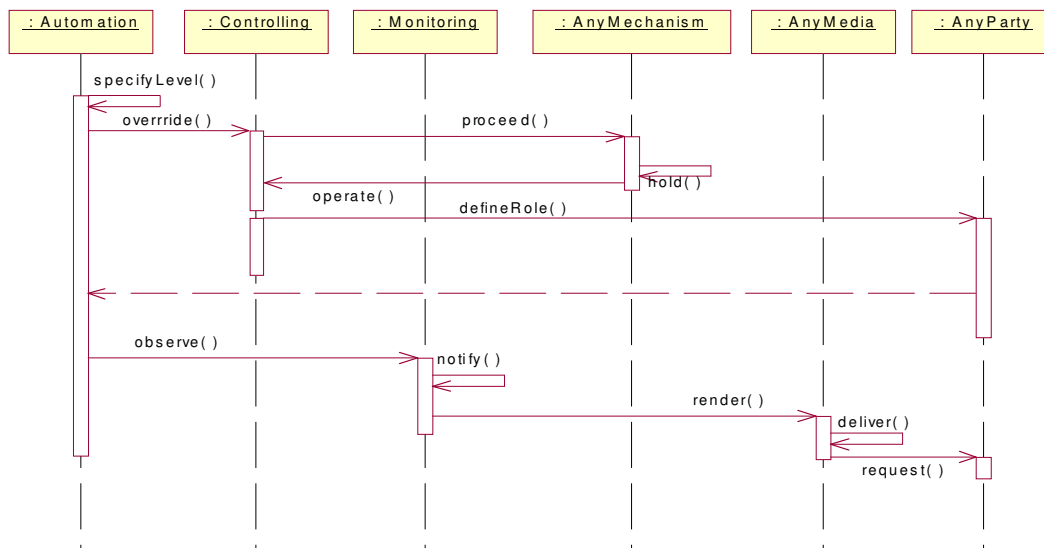


Figure 2: Sequence Diagram of the Automation Analysis Pattern

#### Participants:

#### Classes:

- *Automation*. Represents the automation process itself. This class contains the behavior and properties that define and regulate the levels of automation across

the system functional areas. It also specifies the mission or goals of the automation task, which most of the time correspond to the mission of the system itself or the original system in the case of automation being introduced a posteriori.

- *Monitoring*. Provides access to system state. It is represented through some media to the parties involved in controlling the automation process
- *Controlling*. Is the interface to the mechanism(s) being automated. It regulates and operates on the processes of control in the system. It serves as a proxy to the actual entities being affected.
- *AnyMechanism*. Encompasses a broad representation of any type of process, mechanism, operation or activity that can be automated. It has no restrictions on the types and location of the entities involved.

- *AnyParty*. Represents the automation operators. It models all the parties that are involved in the automation process. Party can be a person, organization, or a group with specific orientation. The pattern diagram and detailed pattern description is provided in [7]
- *AnyMedia*. Represents the media through which monitoring takes place. For instance, a system might employ digital screens to express data on the system status. The pattern diagram and detailed pattern description is provided in [7].

#### CRC Cards:

The CRC card names the class, responsibility, and its collaborations. The CRC card also names a role for each class, which is useful for identifying the class responsibility. Each class should have only one and unique responsibility. The collaboration consists of two parts: clients and server. Clients are classes that collaborate and have relationship with the named class. The Server contains all the services that are provided by the named class to its own clients [7]. A group of CRC Cards representing the Automation Analysis Pattern is showed in Figure 3.

Patterns:

Automation (Automation Handler)		
Responsibility	Collaboration	
Describe the automation concept itself.	<b>Clients</b> Controlling Monitoring	<b>Server</b> specifyLevel ()

Controlling (Controlling Descriptor)		
Responsibility	Collaboration	
Represents the Controlling techniques aided by the Pattern.	<b>Clients</b> AnyParty AnyMechanism	<b>Server</b> override () operate ()

Monitoring (Monitoring Handler)		
Responsibility	Collaboration	
Specify the monitoring actions over particular mechanisms.	<b>Clients</b> AnyMechanism AnyMedia	<b>Server</b> observe () notify ()

AnyMechanism (Mechanism Descriptor)		
Responsibility	Collaboration	
Represents the abstract mechanisms enhanced by the automation process.	<b>Clients</b> Controlling Monitoring	<b>Server</b> proceed () hold ()

AnyMedia (Media)		
Responsibility	Collaboration	
Represents the Media used to assist the Monitoring process.	<b>Clients</b> Monitoring	<b>Server</b> render () deliver ()

AnyParty (Process Supervisor)		
Responsibility	Collaboration	
Supervise the automation process.	<b>Clients</b> Controlling	<b>Server</b> request () defineRole ()

Figure 3: CRC Card Representation of the Automation Analysis Pattern

#### Consequences:

- The use of the Automation Analysis Pattern offers the following benefits:
  1. *Embody the Basis for the Automation Concept per se*: The pattern reflects the ability of a system containing or wishing to introduce automation to exercise control over the level or *degree* and the *type* of automation. For most systems, this represents automation of processes, control, mechanisms and/or data display and interpretation. Implicit in the analysis pattern

then should be the ability to automate any aspect of a complex system. This is achieved by the Stable Automation artifact.

2. *Handles the automation of more than one Mechanism*: The Stable Automation Analysis pattern is general enough to handle distinct mechanisms to be automated, different in nature and process flow. With such flexibility, the automation pattern needs to adjust its properties in order to accomplish a proper automating action. This is done by the use of the AnyMechanism artifact.

#### IV. PATTERN APPLICABILITY

In order to illustrate the use of the Automation pattern in different application areas, two examples are presented: Automating the process of Driving a Car through the utilization of certain mechanisms and built-in equipments such as Navigation system, etc; and the Automation of a Power Plant focusing on the monitoring aspect of the processes, and the context awareness for the operator, etc. Since the purpose of these examples is to demonstrate the usage of the proposed pattern, and for simplicity, these examples do not present the complete model of the problem. Instead, they focus on the part that involves the automation process.

3. *Embody distinct controlling techniques:* The Automation Analysis pattern is abstract enough to embody a plethora of controlling techniques used for distinct types of automated mechanisms. This plethora of techniques is covered by a core abstraction represented by the Controlling Business Object.
  4. *Handles distinct Monitoring techniques:* Distinct from current solutions, the automation pattern encompass as one stable abstraction a large variety of monitoring techniques to assure a proper implementation of automated process or mechanism. This core abstraction is embodied by the stable Monitoring artifact.
  5. *Consider Different Media Types:* The Stable Automation Analysis pattern considers the monitoring of certain mechanisms by different media types. This is accomplished by using the AnyMedia pattern, which represents the media type and its kinds. This feature increases the flexibility of the pattern since the automation problem is incurred in different applications, and the controlling of its mechanism by the use of distinct media (i.e. Expert systems) is an important issue per se.
  6. *Adaptable for Required Application Areas:* The Stable Automation pattern structure maintains a high level of adaptability across different application areas. This automation pattern represents a stable and *reusable* core that can be utilized in different areas that share the same problem domain. The determination of these application areas will adapt the automation pattern to best meet the goals of this automation concern.
- The use of the Stable Automation pattern has the following limitations:
    1. **Lack of Pattern Representation.** At first impression, it would be hard to discover, in a wide sense, the several hidden concerns within the patterns that are included in the stable Automation Analysis pattern. Such as, concerns related to the assignment of roles of the entities performing, controlling, and monitoring an automation problem; however, these concerns should be considered within the AnyMechanism, AnyParty, AnyMedia patterns details.
    2. **No Industrial Objects to Clarify Pattern Applicability.** Since the Stable Automation pattern has been developed based on software stability concepts, there are no IOs attached to the pattern itself, which makes the pattern's applicability not very obvious from just reading the Automation pattern structure. However, attaching such IOs (which are implementation-specific details) will narrow the applicability of the pattern. Showing detailed case studies for the pattern applicability make the pattern usage obvious; yet, preserve the generality of the main pattern.
1. **Scenario 1, Car driving:** Driving a car is one of those tasks that we will eventually like to see being automated, safely and satisfactorily. Efforts are now being put into building the "smart" roads that will be required to make this feasible. Notice in the diagram above, that "Road" is an industrial object, which reflects that very fact, as an application-specific object, which is part of the system being automated. In this case, the mission of the automation process is to get from point A to point B, in time and safely. The automated mechanism is composed of the car itself, but more importantly it's controlling elements, and the road, which is an integral part of the process. The monitoring mechanism in this case is a Navigation System, which is composed of a GPS system, a road map system and a display of the car's status, such as speed, gas, and mileage. The party involved in this scenario is the single driver, which could be actually doing the driving, with some automation support, or, only performing the supervisory tasks, and leaving all the driving to the system. Notice the multiplicity on the "AnyParty" Business Object (BO). It indicates that its relationship to the Controlling class is optional.

This is a type of application similar to other types of navigations, such as plane, ships, and so on. In these other cases, the party involved would be a whole crew of people, and the complexity of analysis much greater, but the pattern would still hold. Another aspect to note, is that the automation Enduring Business Theme (EBT) specifies the level and type of automation that occurs in all areas. That is, in combination with the structure of the pattern relationships, it can determine what and how much is being automated. Figure 4 shows the stable model for this example. Figure 5 shows the respective sequence diagram for this example as well.

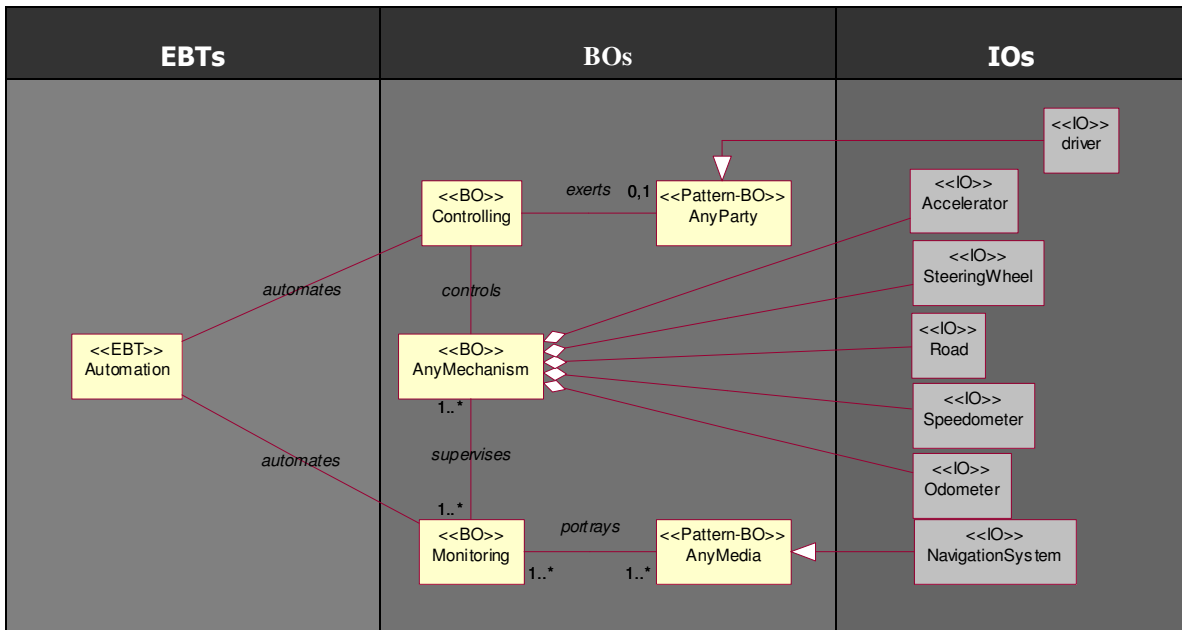


Figure 4: Stable model of the Automation Scenario: Driving a Car

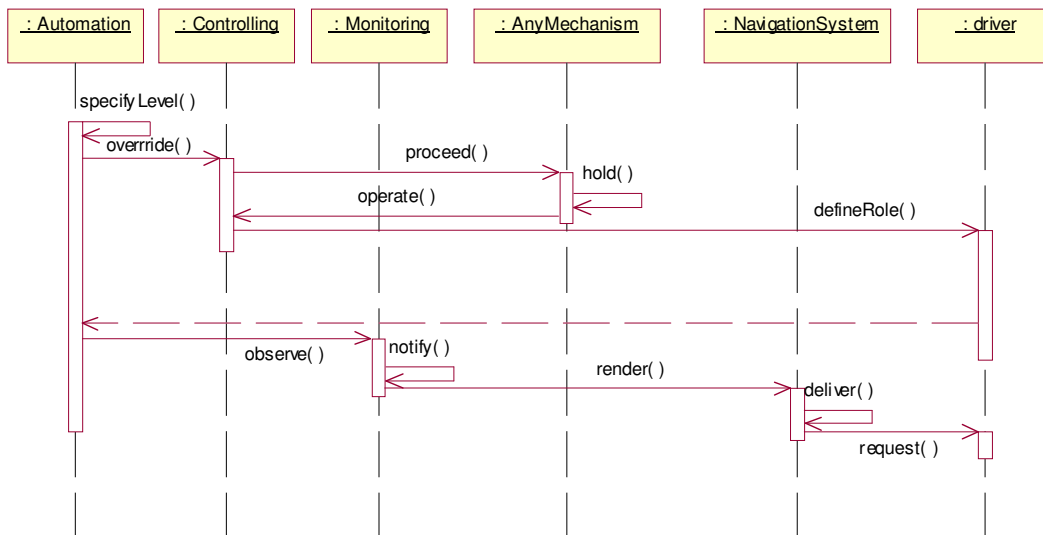


Figure 5: Sequence Diagram of the Automation Scenario: Driving a Car

2. **Scenario 2, Power plant:** The task of automating a power generating plant primarily requires automation to ensure safety of operation. Given the real-time constraints of inherent in the nature of the task, critical decision-making needs to be supported well by automation. In particular, automation in this case focuses on the monitoring aspect of the process, and ideally should provide to the operator, context awareness and good visibility of the system state at all times. The mission of the automation process is then to maintain plant safety, and generating power. To do so, it establishes different levels of automation, more at controlling valves, coolant and charging system, and less so at the decision-making level. Nonetheless,

it provides good feedback at the level of monitoring and provides automation of information processing, such as enabling projection of system states in the future. The plant is composed of a main heater, a coolant system, release and injection valves, and the feed water channel. All of these are Industrial Objects (IO) as shown in the diagram, and are placed at the periphery of the pattern, indicating their application-specific nature, i.e. they can be exchange for other cases. The parties involved are the operators composing the control team. Other IOs are the control board, for manual supervisory control, and the display units, used by the operators to monitor the plant state. Figure 6 illustrates the stable model of this

example. The respective sequence diagram is presented in figure 7.

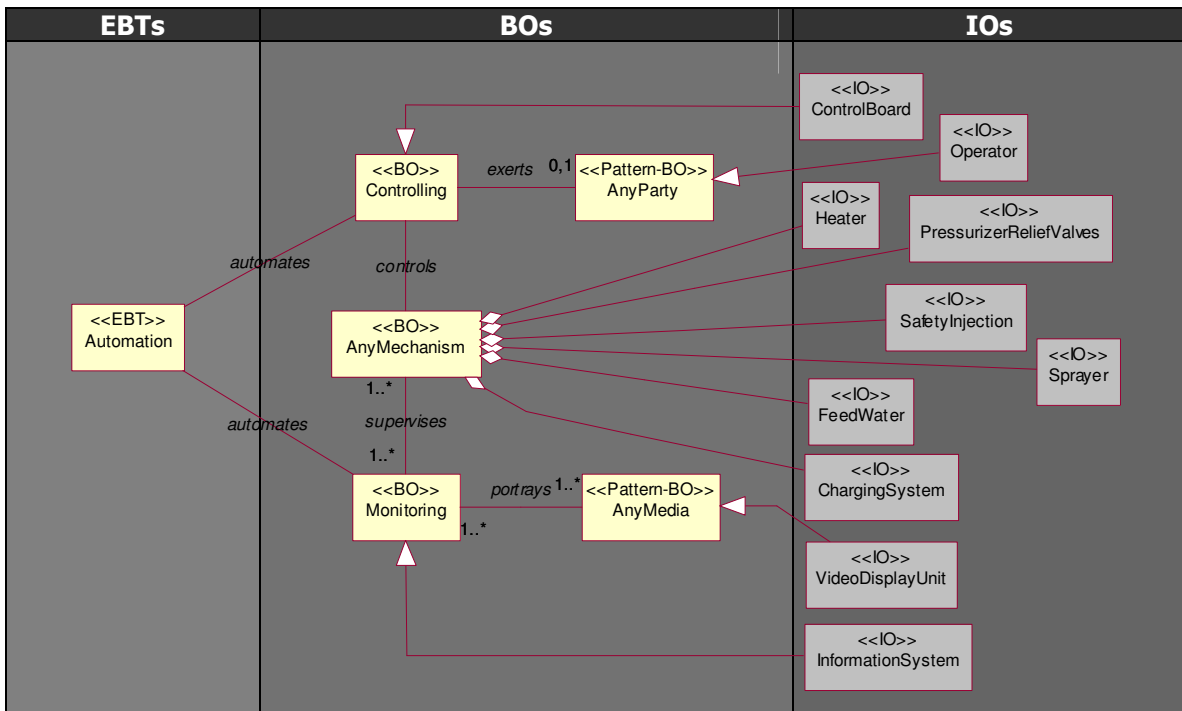


Figure 6: Stable Model of the automation scenario: Power Plant

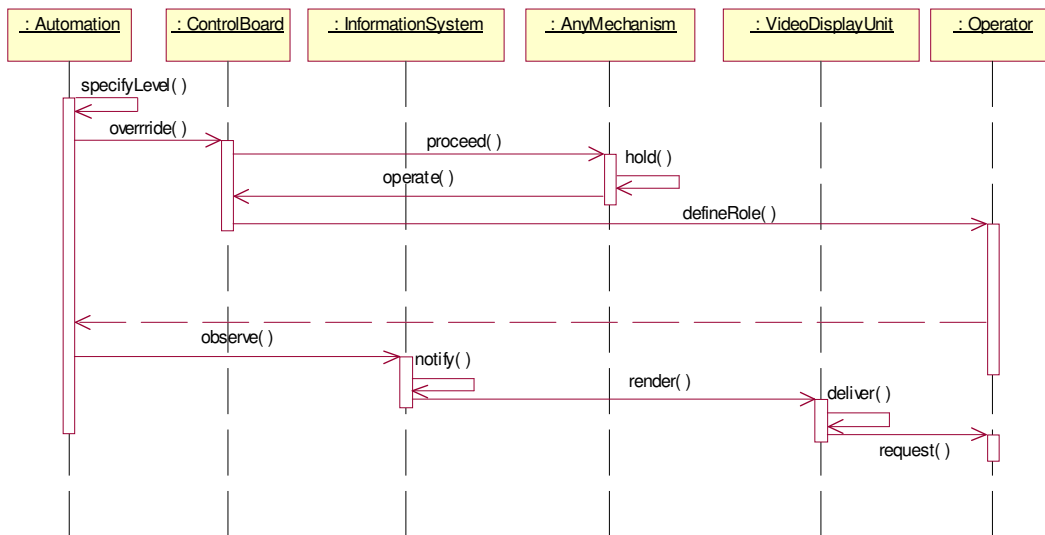


Figure 7: Sequence diagram of the automation scenario: Power Plan

## V. CONCLUSION

The main objective of the work described in this paper is the utilization of the Software Analysis concepts to an application neutral. We have laid down in this paper the beginnings of a stable pattern to represent automation in systems across domains. It lays the foundation for a pattern that will achieve a

true stability model of the concept of automation as it stands for all industries whenever there is the need for introducing or building automation for some process or mechanism. This pattern aims at producing a core abstraction for the concept of automation, which at the most essential level reflects a replacement of some function performed previously by human beings. A valid pattern as this, should also reflect the more

contemporary and broad definition of automation, that is, of controlling and supervising complex systems that are built today and in the future, where human intervention is not necessary at the core level, such as system which are born out of this new age of information technology. Namely, systems which are not replacing a previously existing human task, but that are in effect creating new ones which were never before possible, such as the case of highly complex computational tools, like in Genomics for instance, or the control of distributed networks by means of autonomous agents in the case of B2B infrastructures. Such is the ultimate goal of an automation stability pattern; we have hereby made a humble contribution by putting together a first attempt at bridging these two historical and technological forces. More work would need to be done to validate and tune this pattern so that it applies efficiently to a variety of real world applications.

#### REFERENCES

- [1] Mahdy, Ahmed & Fayad, Mohamed. "A Software Stability Model Pattern". Computer Science and Engineering, University of Nebraska-Lincoln.
- [2] Parasuraman, Raja, Sheridan, Thomas & Wickens, Christopher. "A Model for Types and Levels of Human Interaction with Automation". IEEE Transactions on Systems, Man and Cybernetics. Vol. 30 No. 3 May 2000.
- [3] S.K. Card, T.P. Moran & A. Newell. "The Psychology of Human-Computer Interaction". Hillsdale, New Jersey: Lawrence Erlbaum Associates. 1983.
- [4] D. Brugali and G. Menga. "Architectural Models for Global Automation Systems". IEEE Transactions on Robotics and Automation, Vol. 18, No. 4. August 2002.
- [5] C. Alexander. "A Pattern Language: Towns, Buildings, Construction". Oxford University Press, 1977.
- [6] T. Honkanen. "Design Patterns in Automation". Postgraduate seminar on Information Technology in Automation. Helsinki University of Technology, Information and Computer Systems in Automation. April 2002.
- [7] M.E. Fayad, V. Stanton, and Hamza, H. "A New Look At the CRC Cards." <http://www.activeframeworks.com>
- [8] Miriam-Webster Online Dictionary, <http://www.m-w.com/cgi-bin/dictionary>.
- [9] M.E. Fayad. "Accomplishing Software Stability." Communications of the ACM, Vol. 45, No. 1, January 2002.