The Shell Model – A Method for System Boundary Analysis

Walter Sebron(✉), Hans Tschürtz, and Peter Krebs

Vienna University of Applied Sciences, 1100 Vienna, Austria
{walter.sebron, hans.tschuertz, peter.krebs}@fh-campuswien.ac.at

Abstract. The application of analysis methods in systems and safety engineering depends on the available depth of knowledge about a system in the respective lifecycle phases. However, the analysis method chain shows gaps as it should support system analyses during the whole lifecycle of a system. The aim of this paper is to discuss the Shell Model Analysis method, which aims at closing a gap in early system lifecycle phases, like the concept phase. The Shell Model Analysis is a graphical method that splits up a system and groups its parts into concentric shells, built around a selected system part under consideration. A set of rules and guidelines has been defined in order to enable a proper shell build-up. Originally, the method was intended to assist the analysis of electronic system parts, like e.g. an embedded control unit for a braking system, by supporting the definition and analysis of the system's boundary and its environment only. Meanwhile, it has been extended to also produce results that can be starting points for consecutive analysis methods.

Keywords: Analysis methods · System and safety engineering
System boundary definition

1 Introduction

A variety of different analysis methods is applied in systems and safety engineering, in order to ensure the correct functioning and safety of a system. These analysis methods are usually intended for a specific phase in the engineering lifecycle, depending on what minimal depth of knowledge about the system is necessary.

In [1] (ch. 4.2), the "hazard filter" shows how typical hazard analysis methods – like PHA, HAZOP. FMEA, etc. – are used depending on the according lifecycle phases of a system. The more a system development advances, the more sophisticated the necessary analysis methods become. These methods need deeper details about the system in order to produce meaningful results. Especially the classical (safety) analysis methods (e.g. FMEA, FTA, HAZOP) need detailed knowledge about a system [2] (p. 27). Only the early method types during conceptual design can be applied with a low level of system detail knowledge [1] (ch. 4.3.1).

In order to apply an analysis method to a system correctly, it is necessary to first understand its basic requirements and design [1], (ch. 5.2.2), and to have clearly defined system boundaries [1] (p. 20). Without those boundaries, the system cannot be distinguished exactly from its environment. It also becomes difficult to define the scope of analyses.

System engineering and system safety should look at all aspects of a system as an integrated whole, rather than looking at individual components in isolation from the system [1] (p. 4). This is often termed holistic approach and is also advocated in [2] (p. 19). Though, most, if not all, of the analysis methods applicable at an early lifecycle stage exhibit at least one of the following shortcomings:

- Not following a structured and systematic approach (e.g. brainstorming)
- Not defining the system boundaries (e.g. PHL, PHA)
- Not defining a system structure (e.g. HAZOP)
- Not supporting a holistic view (e.g. FMEA, checklists).

But when analyzing a (sub-)system-to-be-built at a very early stage, it is almost inevitable to come across the need for a structured method that aides in defining the overall system boundaries and its environment, but also allows to focus upon the specific (sub-)system as it is embedded in the overall system.

2 The Basic Idea

2.1 Motivation

Experiences from conducting safety analyses for industrial projects showed several issues, such as:

- different and often divergent viewpoints on the systems-to-be-analyzed,
- incomplete understanding of the boundaries of a system, and therefore of the system environment,
- ambiguities about the newly intended functioning of a system and its components.

This showed the need for a method that supports and emphasizes the understanding of a system as a whole, the definition of its boundaries and environment. Such a method should be applicable already at a very early project stage with only brief knowledge on the system itself and the overall system that it is embedded into. It should facilitate a description of the components and their relationships within the overall system. It should foster awareness of possible environmental influences which may vary due to different usage scenarios. This method should serve as an entry point for further system and safety analyses; its focus on the setting of boundaries of and within a system is key for being able to conduct a variety of analysis methods afterwards.

However, looking into different descriptions [4–6] and collections [1, 3, 7] of analysis methods, no method was found that would satisfy the above mentioned needs, indicating the need for a new method.

2.2 Methodical Background

As stated in [2] (ch. 2.3), when developing a (safety critical) system, understanding of the system, its environment, its workings, its failure modes, etc., is crucial. The main obstacle in this context is the ever-increasing complexity of newly developed systems. Again, according to [2], "[simplicity] is the only real way to ensure understanding, or understandability, of the systems being produced." This simplicity can be achieved by use of appropriate structure. This is why the Shell Model Analysis uses the concept of simplification by focusing on one element of interest, called the "system under consideration" (SUC) and its interactions with other system elements. This way, "unnecessary" details are omitted, thus limiting the efforts for modelling while still allowing purposeful input to subsequent analyses.

Additionally, structuring is achieved by placing the different elements of the overall system into different layers, the so-called "shells." These layers are arranged in a concentric manner; hence a Shell Model (SM) is generated. Furthermore, the structure gets enhanced via displaying the relationships of the different components via arrows, thus defining also their basic interfaces. A preliminary version of the Shell Model Analysis was described in [8], and used as basis for the method defined in this paper.

2.3 Goals of the Method

Since the basic idea for the method was to support the build-up and structure of a system, its boundaries and influencing factors, the main goals of the method are:

- Definition of a system's external boundary
- Depicting the inner structure of a system and the relationship between components
- Consideration of environmental influences
- Definition and consideration of use cases, usage scenarios and situations
- Applicability at a very early lifecycle phase, e.g. pre-concept phase.

Additional goals were identified during development of the Shell Model Analysis method, in order to support, and provide input to, subsequent analysis methods:

- Definition and consideration of the top level functions
- Listing and considering modes of the components and their influences on functions
- Listing of communication types between components and defining their interfaces
- Supporting the discovery and definition of preliminary hazards
- Providing a holistic picture of the system.

3 The Shell Model Analysis Method

3.1 Prerequisites

As the Shell Model Analysis should be applicable at a very early lifecycle stage, little input is required:

- A basic idea or rough concept of the overall system

- The selection of the SUC itself
- Basic functional requirements of the SUC.

3.2 Basic Concepts of the Shell Model Analysis

The Shell Model Analysis represents the structural and functional relationship between the SUC and its surrounding systems graphically as an interconnected structure, organized into a series of concentric stratums, or “shells”. The elements of the structure depict the individual elements of a larger system which communicate with one other to execute functions. In addition, relevant influences from the environment and usage scenarios complement the view. A complete structure represents a system in its environment and is termed a shell model for the system.

Figure 1 depicts the elements of the Shell Model Analysis. A component is a part of a larger system and can be clearly delineated (e.g. a dedicated HW element, such as a sensor or actuator). Each component is identified by an arbitrary component identifier (such as a name or numerical ID) that must be unique over the whole SM.

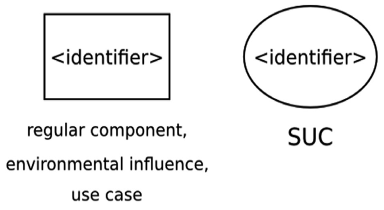


Fig. 1. Elements of the Shell Model Analysis

The SUC is as a special kind of component and exists exactly once in a shell model - hence a unique symbol is reserved for it. Definition of an SUC and its identifier forms the initial step in creating an SM.

In addition to the SUC, a shell model can contain an arbitrary number of regular (i.e. non-SUC) components, or even none at all. The regular components form the “inner” or SUC environment which directly influence or depend on its behavior.

Direct communication between components is symbolized by arrows in between the component symbols for the exchange of messages. Messages include transfer of energy (e.g. electricity) or information (e.g. analogue or digital signals) - either unidirectional or bidirectional. Additionally, regular components can assert an indirect influence on the SUC (e.g. via heat or EMI) which is represented by a “fringe” at the component symbol, including a suitable label. Figure 2 shows the arrows for direct communication and an indirect influence.

Components are grouped into the eponymous shells to convey further information on the logical or physical structure of a system and to facilitate the definition of the analysis scope. Shells are (with one exception) depicted as closed curves of dashed lines which are aligned in a concentric fashion. Each shell must possess a unique and meaningful identifier. Therefore, the Shell Model Analysis shells are non-empty subsets of components (i.e. each shell must contain at least one component) that form a hierarchy, starting from the innermost to the outermost shell.

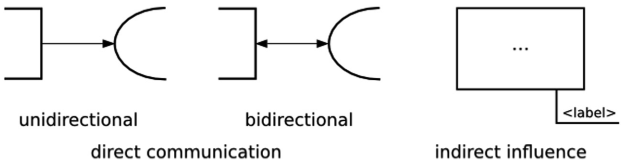


Fig. 2. Arrows for direct communication and notation for indirect influence of component

3.3 Defining Shells

After selection of the SUC, all components that directly communicate with the SUC have to be identified. This includes:

- Components that provide input to the SUC (e.g. sensors, user input elements)
- Components that receive output of the SUC (e.g. actuators, user output elements)
- Components that communicate bidirectionally with the SUC (e.g. clients, servers).

Each of the identified components directly impacts or is impacted by the SUC and is termed an “adjacent component” (AC) of the SUC. The SUC and its ACs are grouped into one shell which is used as the innermost shell of the shell hierarchy as the SM is further developed. As the ACs are situated at a “hop distance” of 0 from the view point of the SUC, this initial shell is termed the zero shell (Z-shell). A well-formed SM must contain exactly one Z-shell which must consist of only the SUC and its ACs linked by respective communication arrows.

Figure 3 shows a typical example of a Z-shell with a generic Programmable Logic Controller (PLC) as SUC, together with its surrounding components (the ACs). Note the communication to the component “Communication Bus”, which is defined as an AC although it could physically extend from the SUC by quite a distance.

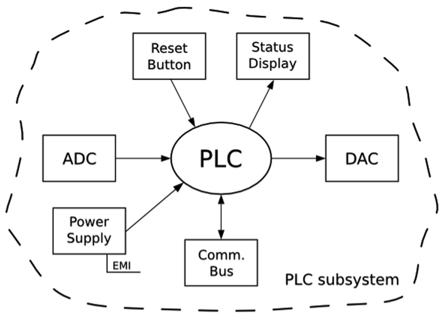


Fig. 3. Example of a zero shell depicting a Programmable Logic Controller (PLC) and its ACs

After definition of the Z-shell, further analysis might identify components that provide input or receive output from the ACs of the SUC, i.e. communicate only indirectly with the SUC. These “next order components” are placed outside the Z-Shell and are grouped together into a new shell that fully encloses the Z-shell. This outer shell (O-shell) of order one (O_1-shell) therefore contains only components with a distance of "one hop" from the SUC.

Figure 4 shows how the running example of the PLC is extended by an O_1-shell containing components that provide/receive indirect input (Sensor) or output (Actuator) to/from the SUC. Note that some ACs might not have a corresponding component in the O_1-shell.

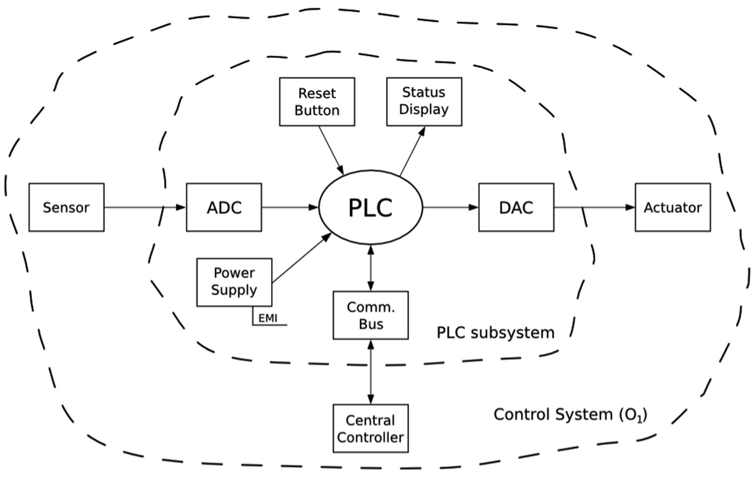


Fig. 4. Adding the first O-shell with indirectly communicating components

The principle of identifying indirect senders/receivers and adding them by means of outer shells can be repeated an arbitrary number of times. This results in a hierarchy of shells with increasing order. The amount of O-shells depends mainly on the complexity of the analyzed system and the intended scope of the analysis. After adding the last O-shell, this outermost O-shell is designated as the border of the system and is termed the system border shell (S-shell). To facilitate visual recognition, the border of the S-shell is drawn with a solid line.

Figure 5 shows the addition of a second O-shell (O_2-shell) with a component "Operator" representing the user interacting via some user interface elements with the SUC. As the construction should end at this point, the O_2-shell is marked as S-shell by its solid line. Of particular note is that the "Operator" communicates directly with two ACs, effectively "skipping" the O_1-shell. The order of an O-shell should therefore not be conflated with the "hop order" of its components. Note however, that only O-shells might be skipped in this way - direct communication between the SUC and a component of an O-shell is not allowed.

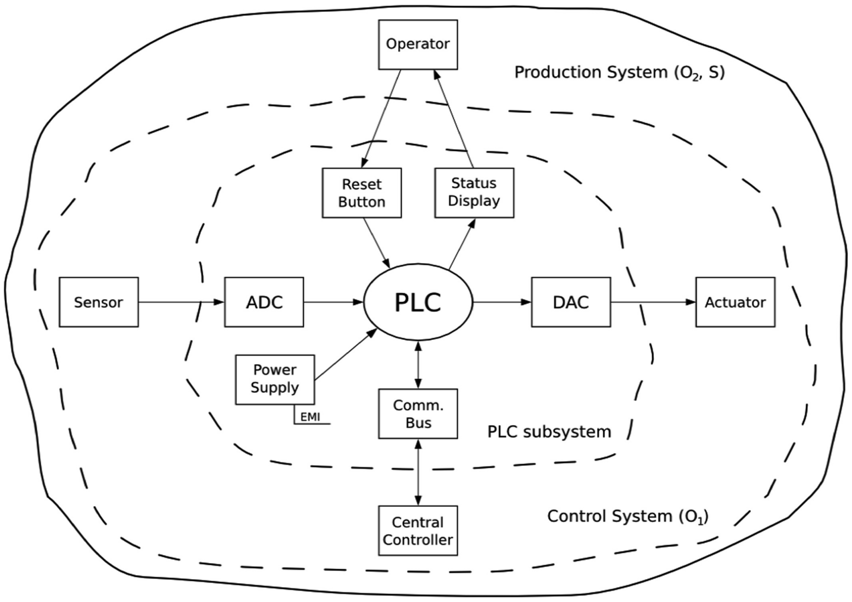


Fig. 5. Adding a second O-shell (O_2-shell) as the S-shell of the example system

3.4 Further Shells

The shells contained within the system border describe the structure of the analyzed system and can be considered as an "internal" environment for the SUC. Information on the environment external to the system is added via two specialized shells.

The first one, the environment shell (E-shell) encompasses the S-shell and contains elements that represent environmental influences that might plausibly affect the SUC or regular components. A typical example is the influence of high or low temperature on the performance of electronic components. Environmental influences differ from the indirect influences exerted by regular components in that they originate from outside the system and are not limited to E/E/PE-related phenomena. Each environmental influence is identified by a suitable identifier (e.g. "heat", "moisture", "vibration").

The second one, the use case shell (U-shell) encircles the E-shell and contains use case elements that represent usage scenarios or situations that are relevant to the overall system and the SUC (usually during operation). The intention is to define a set of disjoint "profiles" for further analyses, which can be used to analyze the SUC under different circumstances (e.g. low vs. high demand).

A complete SM for the running example is depicted in Fig. 6. Note that no communication arrows originate or end at elements of the E- and U-shell.

3.5 Complementary Information

The information conveyed by the elements of an SM is, in general, not enough to fully understand the workings of a system. This is deliberate in order to avoid overloading

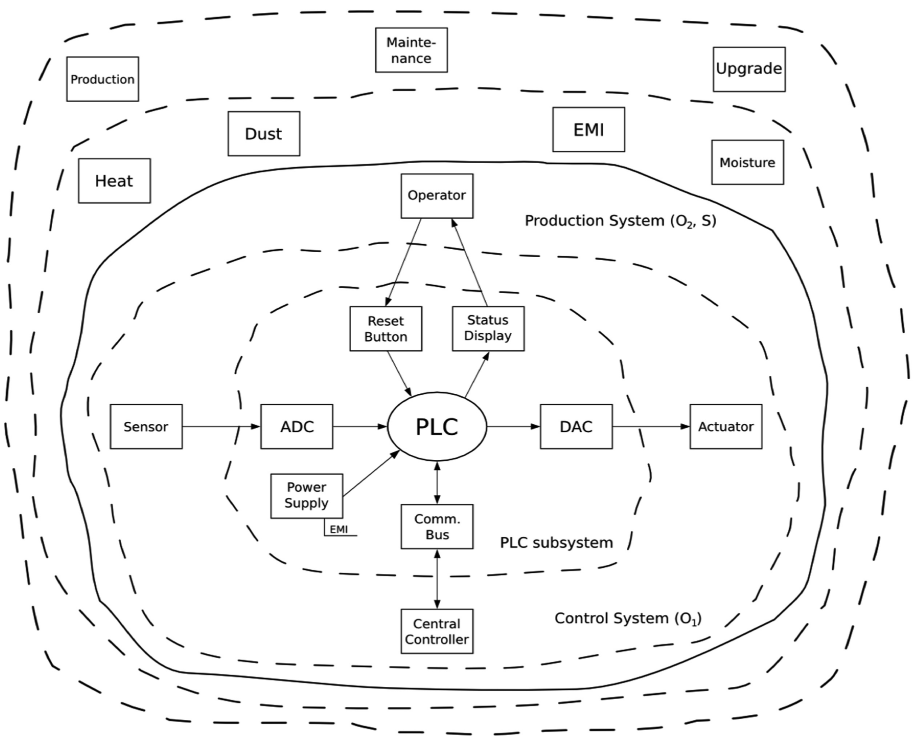
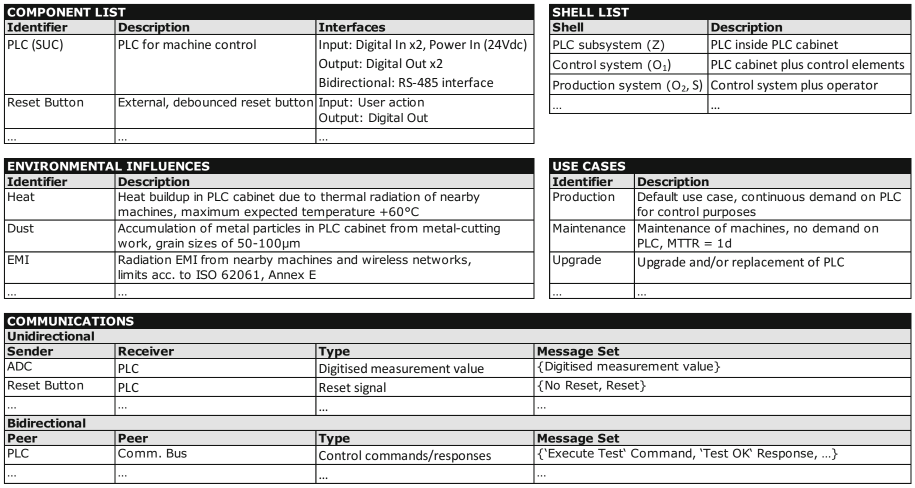


Fig. 6. Complete PLC example with added E-shell and U-shell

the graphical depiction. However, for a thorough understanding of the system, additional information is necessary. Specifically, the following complementary data should be specified for an SM:

- For each component, including the SUC, a description of the component's purpose in the system and its interfaces for direct communication
- For each regular component with indirect influences, a description of each indirect influence asserted by the component and detailed constraints (e.g. exact thresholds for temperature, EM field strength, etc.)
- For each communication arrow, the set of messages that can be exchanged between the communicating components
- For the Z-shell and each O-shell, a description of the shell's meaning (i.e. the criterion for choosing this shell)
- For each environmental influence, detailed constraints similar to indirect influences
- For each use case, a description focusing on important influences on the system and SUC (e.g. expected demand rate).

The formal representation of the complementary information is not prescribed by the Shell Model Analysis. One possibility is to use a "data dictionary" - basically a set of simple tables with references to the elements of an SM. An excerpt of the data dictionary for the running example can be seen in Fig. 7. It is suggested to use a configuration management system for maintaining an SM and its associated data dictionary.



COMPONENT LIST		
Identifier	**Description**	**Interfaces**
PLC (SUC)	PLC for machine control	Input: Digital In x2, Power In (24Vdc) Output: Digital Out x2 Bidirectional: RS-485 interface
Reset Button	External, debounced reset button	Input: User action Output: Digital Out
…	…	…

SHELL LIST	
Shell	**Description**
PLC subsystem (Z)	PLC inside PLC cabinet
Control system (O_1)	PLC cabinet plus control elements
Production system (O_2, S)	Control system plus operator
…	…

ENVIRONMENTAL INFLUENCES	
Identifier	**Description**
Heat	Heat buildup in PLC cabinet due to thermal radiation of nearby machines, maximum expected temperature +60°C
Dust	Accumulation of metal particles in PLC cabinet from metal-cutting work, grain sizes of 50-100µm
EMI	Radiation EMI from nearby machines and wireless networks, limits acc. to ISO 62061, Annex E
…	…

USE CASES	
Identifier	**Description**
Production	Default use case, continuous demand on PLC for control purposes
Maintenance	Maintenance of machines, no demand on PLC, MTTR = 1d
Upgrade	Upgrade and/or replacement of PLC
…	…

COMMUNICATIONS			
Unidirectional			
Sender	**Receiver**	**Type**	**Message Set**
ADC	PLC	Digitised measurement value	{Digitised measurement value}
Reset Button	PLC	Reset signal	{No Reset, Reset}
…	…	…	…
Bidirectional			
Peer	**Peer**	**Type**	**Message Set**
PLC	Comm. Bus	Control commands/responses	{'Execute Test' Command, 'Test OK' Response, …}
…	…	…	…

Fig. 7. Excerpt tables of the data dictionary for the example in Fig. 6.

3.6 Derivation of System Functions

A shell model is a static depiction of a system's structure. However, it is possible to describe the dynamic behavior from the relationships between communicating components. This allows further analyses of a system's intended and, more importantly, unintended behavior.

A system function (or simply, function) is defined as an alternating sequence of components and unidirectional communications between them. The most basic function consists of one component, the sender, communicating some energy or information to another component, the receiver. Obviously, only components linked by a communication arrow in the SM can be part of a function and the communication must match the direction and definition of the arrow. The communication between sender and receiver can consist of a non-empty subset of messages defined for the communication arrow. By prefixing a proper message subset with a '!', the subset consisting of the set difference between the given subset and the complete set of messages for this arrow can be stated for convenience. In addition, a '*' symbol represents the complete set of messages.

In order to specify changes in the status of a component in response to communications or as part of a function execution, each component is augmented by a mode. To symbolize an internal mode change, a dashed arrow is used that links two instances of the same component with differing modes.

Figure 8 depicts a basic function (a) and an internal mode change (b) in a graphical and sequence-based notation, where the latter represents a function/mode change as a sequence of (component, mode) tuples and {message} subsets. By chaining basic functions and mode changes, complete functions over the components of an SM can be defined. The component at the start of the chain is termed "trigger", the component at the end of the chain is termed "terminator" of a function. Thus a complete function (from trigger to terminator) is represented via a sequence of functional steps and messages, a "function-message chain" (FMC).

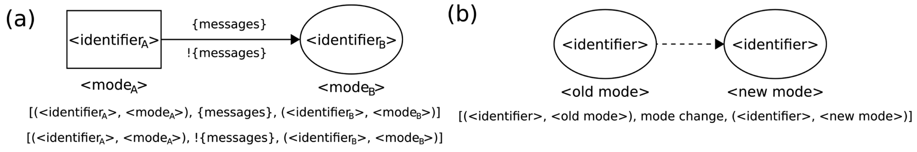

(a)
<identifier_A>
{messages}
!{messages}
<identifier_B>
<mode_A>
<mode_B>
[(<identifier_A>, <mode_A>), {messages}, (<identifier_B>, <mode_B>)]
[(<identifier_A>, <mode_A>), !{messages}, (<identifier_B>, <mode_B>)]
(b)
<identifier>
<identifier>
<old mode>
<new mode>
[(<identifier>, <old mode>), mode change, (<identifier>, <new mode>)]

Fig. 8. (a) Basic function and (b) internal mode change in graphical and sequence notation

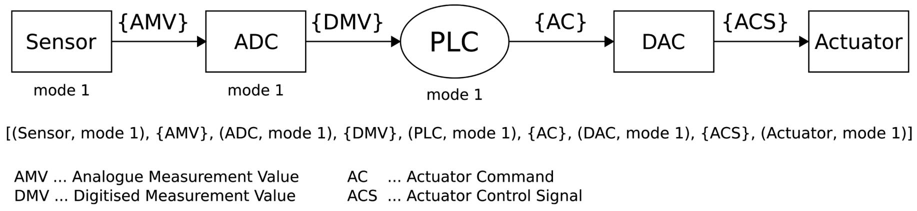

Sensor
{AMV}
ADC
{DMV}
PLC
{AC}
DAC
{ACS}
Actuator
mode 1
mode 1
mode 1
[(Sensor, mode 1), {AMV}, (ADC, mode 1), {DMV}, (PLC, mode 1), {AC}, (DAC, mode 1), {ACS}, (Actuator, mode 1)]
AMV ... Analogue Measurement Value
DMV ... Digitised Measurement Value
AC ... Actuator Command
ACS ... Actuator Control Signal

Fig. 9. A simple example function for the PLC shell model

Figure 9 shows the FMC for a simple example function for the "PLC" SM with the components "Sensor" and "Actuator" as the trigger and terminator, respectively. Note that the mode is omitted for some components - in this case, the component is understood to assume its default mode which must be defined for each component.

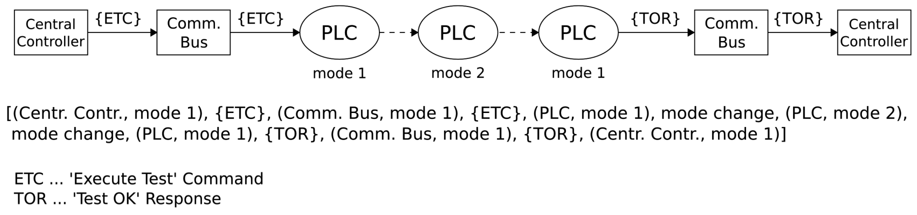

Central Controller
{ETC}
Comm. Bus
{ETC}
PLC
PLC
PLC
{TOR}
Comm. Bus
{TOR}
Central Controller
mode 1
mode 2
mode 1
[(Centr. Contr., mode 1), {ETC}, (Comm. Bus, mode 1), {ETC}, (PLC, mode 1), mode change, (PLC, mode 2), mode change, (PLC, mode 1), {TOR}, (Comm. Bus, mode 1), {TOR}, (Centr. Contr., mode 1)]
ETC ... 'Execute Test' Command
TOR ... 'Test OK' Response

Fig. 10. Example PLC function with an internal mode change of the SUC

A more elaborate example function is depicted in Fig. 10. Here, the “PLC” undergoes a mode change in response to a communication from the “Central Controller”, reverting to the original mode afterwards (supposedly after executing the command). Note that only the order of function steps can be modelled - the real time behavior of functions is out of scope of the notation.

4 Modelling Guidelines

Following the syntactical rules of the Shell Model Analysis leads to well-formed shell models but does not guarantee maximum readability and suitability for further analyses. In the following, a sample of important aspects and advice concerning the construction of practically usable shell models is given.

4.1 Placement of ACs

In many cases, ACs assume clearly defined roles in their communication with the SUC. By consistently placing ACs according to their role at certain positions relative to the SUC, the amount and ratio of interfaces of the SUC can be easily discerned. The following placement is suggested:

- ACs which only provide input, to the left of the SUC (e.g. sensors)
- ACs which only receive output, to the right of the SUC (e.g. actuators)
- ACs which represent user interface elements, to the top of the SUC (e.g. buttons)
- ACs which communicate bidirectionally with the SUC, to the bottom (e.g. busses).

4.2 Functional Relevance of Communication

When adding communication arrows, the choice should be guided by the relevance of the communication for subsequent analyses. A principle rule is that only communication paths that include the SUC should be added. This excludes arrows between regular components that are not a part of a function where the SUC is involved.

For functions where the SUC is included, the scope of communication must be limited to a reasonable amount in order to balance completeness of the system view against complexity of further analyses. For example, a communication between the “Power Supply” and the “Sensor” could be added to the “PLC” example and the derived function in Fig. 9. This would allow an analysis to consider failures of the power supply and its consequences on the function.

4.3 Logical Versus Structural Definition of Shells

Shells can be defined in a variety of ways, with the “ideal” choice depending on the system and the choice of subsequent analyses. Often, the most suitable choice is based on logical distance of regular components to the SUC. In this case, an outer shell’s order is synonymous with the amount of components that must be traversed over communication arrows in order to reach the SUC. The O_1-shell in the “PLC” example was defined in this way.

An alternative to logical order is to define shells according to a system's physical layout. In this case, shells of higher order contain components which are situated at a further physical distance from the SUC. By aligning shells with physical demarcations (e.g. containments, fire barriers), analyses regarding the impact of environmental influences on the system components can be facilitated.

5 Conclusion and Further Considerations

As the Shell Model Analysis is representing a method at a very early lifecycle stage, its findings and results can be used as inputs for further analysis methods. This way, the Shell Model Analysis closes the method gap at this stage, providing input to e.g. a HAZOP Analysis, an FMEA, a qualitative FTA (limited to single component level), a further requirements analysis, a use-case and scenario analysis, and further functional analyses.

The Shell Model Analysis only supports qualitative analysis, as quantification is infeasible at such an early lifecycle stage. Also, the Shell Model Analysis is not suitable for describing or handling parallel structures in the sense of redundant solutions. As of now, the Shell Model Analysis is restricted to electrical, electronical, and programmable electronic (E/E/PE) systems.

The application of the Shell Model Analysis to technologies other than E/E/PE systems (such as mechanics, hydraulics, and pneumatics) and other extensions to the method are under investigation. Furthermore, the method would clearly benefit from a tool or tool chain supporting the graphical modelling effort, the definition of the data dictionary, the derivation of the FMCs (graphical and/or tupel notation), and even parsing the FMC tupels of the FMCs for ambiguities or other discrepancies.

References

1. Ericson II, C.A.: Hazard Analysis Techniques for System Safety, 2nd edn. Wiley, New Jersey (2016). ch. 5.2.2
2. McDermid, J.: Issues in Development of Safety-Critical Systems, Safety-Critical Systems, First ed., pp. 16–42. Chapman & Hall, London (1993)
3. IEC: IEC 61508-6 Functional safety of electrical/electronic/programmable electronic safety-related systems, ed. 2.0, part 6. IEC, Geneva (2010)
4. IEC: IEC 60812 Analysis techniques for system reliability – Procedure for failure mode and effects analysis (FMEA), ed. 2.0. IEC, Geneva (2006)
5. U.S. Nuclear Regulatory Commission: NUREG-0492 - Fault Tree Handbook. U.S. Government Printing Office, Washington (1981)
6. Ericson II, C.A.: Fault Tree Analysis Primer, 2nd edn. CreateSpace Independent Publishing Platform, Charleston (2011)
7. Preiss, R.: Methoden der Risikoanalyse in der Technik. Edition TÜV Austria, Vienna (2009)
8. Tschürtz, H.: Safety-Vorgehensmodell zur Konzeption und Entwicklung von sicherheitskritischen Systemen. Dtechn Thesis, Institute for Engineering Design and Logistics Engineering, Vienna University of Technology, Austria (2016)