System Engineering for dependency analysis - a Bayesian approach: application to obsolescence study

Amel Soltan, Sid-Ali Addouche, Marc Zolghadri, Maher Barkallah, Mohamed Haddar

Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 84 (2019) 774–782

www.elsevier.com/locate/procedia

29th CIRP Design 2019 (CIRP Design 2019)

System Engineering for dependency analysis - a Bayesian approach: application to obsolescence study

Amel Soltan[a,b,*], Sid-Ali Addouche[a], Marc Zolghadri[a], Maher Barkallah[b], Mohamed Haddar[b]

[a]Quartz laboratory, SUPMECA, 3 rue Fernand Hainaut, 93407, Saint-Ouen, France
[b]Ecole Nationale d'Ingnieurs de Sfax, Universit de Sfax, L2MP, Route Soukra Km 3.5, BP 1173, 3038 Sfax, Tunisia

Abstract

Throughout its life cycle, systems undergo several modifications in their architecture. These changes target at remaining competitive and responding quickly to new customer requirements. However, any entity change (i.e. component, function or functionality) can produce unexpected consequences, propagated throughout the whole system architecture. It is then necessary to model, predict and control them. System engineering tools and techniques allow dealing with complex systems design. That is why we have developed a novel methodology to analyze changes using a system engineering methodology called ARCADIA, developed by Thales and its associated software Capella. The obtained models allow mapping various kinds of dependencies within a system architecture. The method, presented in this paper, shows how these models are used to integrate change propagation and transform them into Bayesian networks. A set of experiments allows then to obtain insightful pieces of knowledge about the changes propagation. An illustrative case is developed with a focus of particular changes caused by obsolescence of component, function or functionality.

Keywords: Architecture ; System Engineering ; Change propagation ; Bayesian network ; Obsolescence

1. Introduction

Companies accelerate the pace of engineering changes in their products and systems to stick with new requirements. They change functions, components or functionalities (the offered services to customers) of the product to keep their market share. This accelerated pace generates obsolescence and new products should replace the old ones. The engineering changes affect the entities that make up the product architecture (component, function, functionality) and the dependencies between them (inter-components, inter-functions, function-components, etc.). In this article, we focus on the resilience of products to changes and especially those related to obsolescence. Therefore, the very first step, after the identification of components and functions, is to explicit dependencies among them by using the system architecture modelling. This work demonstrates the importance of using system engineering to provide a solid foundation for engineering change studies. We chose one of the most well established system engineering methodologies, ARCADIA developed by Thales, because among others, ARCADIA provides a clear roadmap for system modeling at various levels. Moreover, it is supported by an open source modelling environment Capella [13]. Capella allows controlling the validity of the models through various modelling steps. It provides also possibilities to extract data required to analyze engineering changes propagation. Based on system models, we transform them into dependency models. They will be the basis of the analysis using Bayesian networks to study the impact of changes within the architecture. The paper is organized as follows. In the next section, we will develop a literature review. In the third section, we propose a methodology called "a System Engineering based Change Engineering Methodology, noted SE-CEM, to model, analyze, and predict change propagation in a system architecture. The fourth section contains an application of this methodology thanks to a case study "EOLE" borrowed from [13].

* Corresponding author - Tel.: +33-65-991-4059
E-mail address: amel.soltane@supmeca.fr (Amel Soltan).

2. State of the art

2.1. Obsolescence

Obsolescence is a general term, nowadays frequently used, whose common definition is "The state of being which occurs when an object, service or practice is no longer wanted even though it may still be in good working order", [7]. In a more precise manner, a product or system is obsolete when it no longer meets the functionality expected by customers. No product or system is immune from obsolescence. The causes of obsolescence are multiple. Authors in [3] list some of them: technology advancements, no vendor support, competition, mergers and acquisitions, environmental requirement.

It is crucial to mitigate the risk of obsolescence to minimize costs throughout the product lifecycle. There are several mitigation strategies. Sandborn [14] defines 3 types of obsolescence management: reactive, proactive and strategic. The reactive management is made for unexpected obsolescence events; a quick and immediate solution should be found. The proactive obsolescence management makes decisions when the obsolescence has not yet been produced, but its possible date is foreseen. In this sense, Solomon et al. [16] proposed a forecasting approach. Finally, the strategic obsolescence management looks for reducing the risks of obsolescence by using different pieces of data and knowledge (date of obsolescence, sales forecasts and logistics data) from the very first steps of system development.

2.2. System Engineering (SE)

According to EIA-ISO 632 [17], "System Engineering is an interdisciplinary approach that encompasses all technical efforts to develop and verify a set of system, user, and process solutions in a total, integrated lifecycle to meet customer needs. System engineering (SE) methods are used for modelling complex systems.

To face the complex system design and to improve quality and productivity, SE is transforming into a model-based approach (Model Based System Engineering MBSE) [15]. The use of models helps to provide a source of information used by the development team. They facilitates the integration of system design and specifications.

System Engineering provides an approach based on generic processes and a set of concepts (function, scenario, system element, requirement, *etc*.). A process describes *what to do* and it is composed of a set of activities and tasks organized around a purpose. To describe the *how to do*, we use modeling techniques and methods. The activities and tasks of SE are transformations of generic data called entities. Each entity is characterized by specific attributes that can have different values. To follow a logical sequence of operations, one must explicit relationships between entities. SE ontology is a set of entities and their relationships. This ontology provides a lot of benefits (1) standardized vocabulary used in different modeling methods and technique (2) the identification of the impacts of modifications in the System Engineering entities [15]. There are various SE methodologies. ARCADIA is a model-based engineering methodology for the architectural design of complex systems. It appeared in 2007 as part of the Thales Airborne Systems [12]. ARCADIA is supported by Capella [1]. It contains a graphical modeling workbench [10]. Thales created a method that performs functional analysis and the association of functions with the components of the architecture while defining implicitly a modeling language [10]. ARCADIA together with Capella, use the system modeling language, largely inspired by UML and SysML. They allow obtaining a set of knowledgeable models.

ARCADIA contains five levels. For each level, Capella offers a number of diagrams. Although ARCADIA is a structured modeling methodology, it remains flexible. One may create diagrams in "any" order and stop modeling at any time according to the awaited level of understanding. Table 1 gives an overview of the levels and the most important Capella diagrams.

ARCADIA levels	Role and Notations	Capella Diagrams
Operational Analysis	The first step describes the future users of the system called ***Operational Entities***; for humans, we call them ***Operational Actors***. Needs, expectations and objectives are also defined. This step describes what the future system needs to offer to users; ***Operational Capability***.	- **OCB** (Operational Capabilities Blank): This diagram defines the operational capacities, entities and actors, and their relationships. - **OAIB** (Operational Activity Interaction Blank): For each operational capability, system-users interactions are defined. - **OAB** (Operational Architecture Blank): it maps entities to functions.
System Analysis	The focus is put on ***system*** itself: i.e. how the system works to meet the user's needs. These are ***system functions***. ***Functional chains*** link functions together to perform an action	- **SDFB** (System Data Flow Diagram): Through this diagram, we add the functions of the system. - **SAB** (Architecture Blank diagram): It defines the functions, defined in SDFB, performed by the system. The interactions between the system and users defined in the operational level are also shown.
Logical Architecture	The system ***subsystems*** (called ***logical components***) are determined. Functions developed in the System Analysis level are assigned to various subsystems. Their inter-dependencies are explicitly modelled.	- **LAB** (Logical Architecture diagram): System parts and their properties are determined.
Physical Architecture	It defines the system final architecture by adding the ***physical components***. A ***behavior physical component*** executes some of system functions. A ***node physical component*** hosts a number of behavior physical components	- **PDFB** (Physical Data Flow Blank): It contains data exchanges among components. - **PAB** (Physical Architecture Blank): It defines the mapping between the physical components and functions, and contains also the exchanges between these components.
EPBS (End Product Breakdown Structure)	The physical components are grouped into Configuration Items which define the choices to be made under design constraints (Software, Hardware, System, COTS ...)	- **CIBD** (Configuration Items in a breakdown diagram): It defines the tree of the chosen configuration items.

Table 1. ARCADIA Levels

2.3. Engineering change (EC)

Huang and Mak [10] define the engineering change, EC, as the changes and modifications in forms, fits, materials, dimensions, functions, etc. of a product or a component . According to

[1] an Open Source MBSE tool (https://www.polarsys.org/capella/)

Jarratt [11], the Engineering change management (ECM)refers to the organization and control of the process of making alterations to products. The goal of ECM is not only to reduce the number of [necessary] changes, but also to manage those changes, if they are made to reduce losses in time, cost, and quality [18].

A change in an architectural entity may spread to other entities that have direct or indirect dependencies with it. There exist tools to study changes and their propagation by determining the dependencies within the system. Among these tools, the Design Structure Matrix (DSM) represents the dependencies between elements in the same domain using a square matrix. To map between two different domains, Domain Mapping Matrices (DMM) are used. It is a rectangular matrix that contains entries of two DSMs [6]. Another matrix proposed by Gorbea et al. [8], which called MDM (Multiple Domain Matrix). It is the fusion of DSM and DMM, see figure 1.

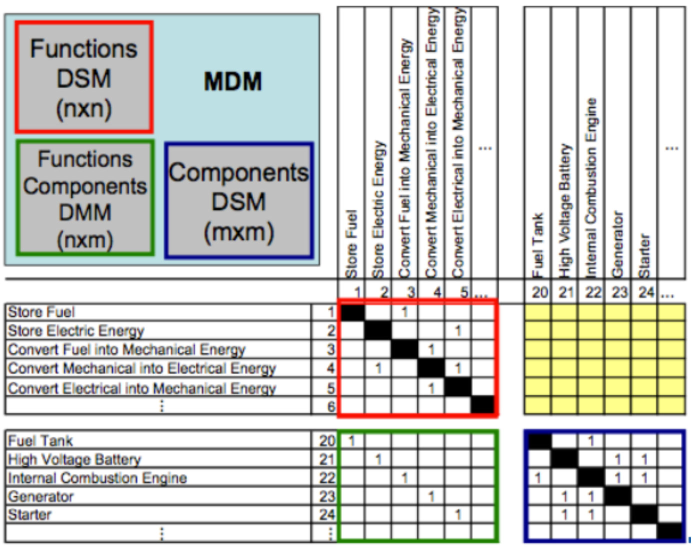


Fig. 1. the Design Structure Matrix (DSM), Domain Mapping Matrices (DMM) and (Multiple Domain Matrix (MDM)[9]

To manage the propagation of changes, Cohen et al. (2000) [5] attempt to analyze the possible consequences of changes through pro-change representation (C-FAR). This method determines the interactions between two different parts of the system in a C-FAR matrix. So, to bind the dependencies throughout the system, one should make calculations by multiplying these matrices [6]. Nevertheless, this method does not suitable for complex systems because the initial requirement which is the dependency identification does not address in a structured manner.

Ollinger and Stahovich [1] proposes in 2001, a computer program (RedesignIT) that models a product as a graph that contains quantities, constraints (FIXED, MAXIMIZE, MINIMIZE) and causal relations (M +, M-). But the results remain abstract because they indicate directions and quantities and do not specify exact numerical values of parameters. There is another method developed by (Clarkson et al., 2004) [4]. It is based on DSMs to study both likelihoods and impacts in the system. These matrices are combined using a route counting algorithm to calculate the risk of propagation.

2.4. *Bayesian Networks*

A Bayesian Network (BN) is a probabilistic graphical method used in causal modeling and probabilistic inference. It is based on a directed acyclic graph (DAG) that represents the conditional independence relations between random variables. To quantify the effect of parents on the variable, a BN uses conditional probability distributions. The states of a node are called modalities for each a probability (marginal or conditioned by the states of the parent nodes) should be provided. The computations are based on the Bayes theorem. For instance, for a parent node A with two modalities $a, \neg a$, and its child node B with two modalities $b, \neg b$, knowing $P(a)$ and $P(\neg a)$, the following values are computed: $P(B = b|A = a), P(B = \neg b|A = a), P(B = b|A = \neg a), P(B = \neg b|A = \neg a)$.

BayesiaLab is one of the most user-friendly BN support tools [2]. It is a powerful tool for applying and practicing Bayesian networks by offering the possibility of diagnosis, simulation and modeling of optimization problems.

2.5. *Research motivations*

If a component becomes obsolete, then components that have direct or indirect relationships with it, can be impacted. In addition, each component in the architecture performs a set of functions. Then, these functions may become obsolete too. A functionality is a mapping between the components and functions to serve the needs of customers. In its turn, it can also be affected by obsolescence. On the contrary, if obsolescence affects a function, it could have impacts on the component(s) which is(are) in charge of its run, and also on the functionalities to which they contribute. The obsolescence of functionalities is often generated by customers or external environment (standards, rules and regulations, laws, etc.). So the functions and components that offer them to the customer may be impacted too. In short, when obsolescence hits an entity of the architecture (component, function, functionality), there is a risk of propagation throughout the architecture. It is, therefore, necessary to be able to limit this spread of propagations and their effects on the whole system. To be effective, any proactive mitigation strategy should be based on the system architecture ; i.e. being able to identify the most sensitive entities and to predict any obsolescence propagation. These are the challenges the research work reported partly in this paper.

3. SE-CEM applied to obsolescence mitigation

We propose a System Engineering based Change Engineering Methodology, called SE-CEM, applied to obsolescence mitigation composed of four steps, see figure 2. These phases are discussed hereafter.

Phase 1: System Modelling. In order to study the propagation of obsolescence, we must model a system in detail.

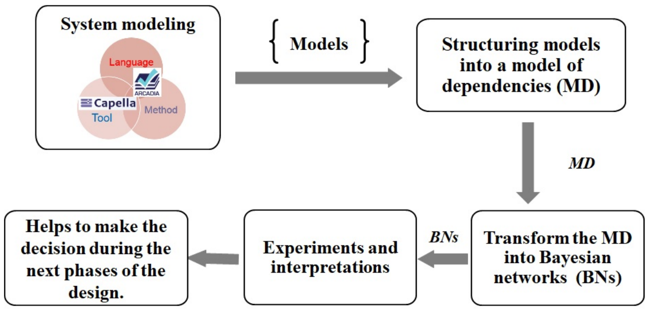

System modeling
Language
Capella Tool
Method
Models
Structuring models into a model of dependencies (MD)
MD
Transform the MD into Bayesian networks (BNs)
BNs
Experiments and interpretations
Helps to make the decision during the next phases of the design.

Fig. 2. Methodology Approach

In this step, we determine the entities that constitute a system (functions, components, functionalities) and the dependencies between them (the relations between functions, the relations between components and the relations between functions and components). In ARCADIA terminology, the functionalities are the operational capabilities, see Figure.3. And to carry this modelling out, functions are defined in the operational analysis phase. To know the system functions, they are specified in the system analysis level. In order to divide the system into sub-systems, the third level of ARCADIA is used as the logical architecture. Finally, the mapping of functions and physical components defines the physical architecture.

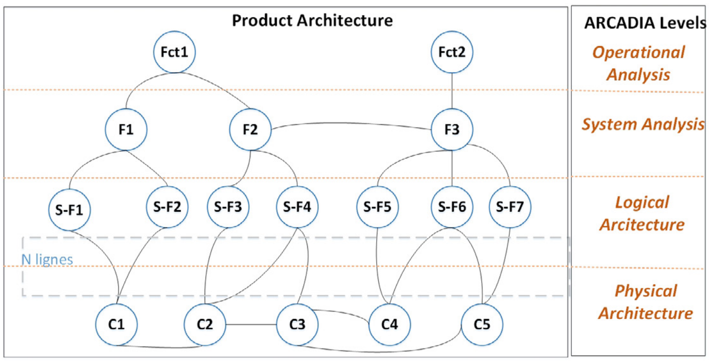


Fig. 3. Product architecture by ARCADIA

Phase 2: Dependency Model (DM). After the system modeling, the goal is to represent the dependencies between the entities of the system by graphs or matrices. This is done by exploiting data from the first step and mapping all the possible dependencies.

Phase 3: Transformation of DM into Bayesian networks To study the impact of propagation of changes, we use the data extracted from the previous step (the dependency matrix). We build Bayesian networks to evaluate the consequences of changes to various entities of the architecture. For example, if a function of a physical component becomes obsolete, we would like to know the propagation of this obsolescence in the system. The question is then to calculate the probability of such change. For instance, if two entities *B depends on A* (the nano-computer which receives data from a pressure sensor), we should compute the conditional probability of "*B becomes obsolete*" knowing the probability of "*A becomes obsolete*". The transformation of a dependency model to Bayesian networks is performed using the rules, which constitute the heart of the SE-CEM. Comprehensive translation rules is given in table.2 and illustrated in section.4.

Phase 4: Experiments and interpretations. After the impact analysis, we may predict the severity and the risks of any potential change attributed to an entity. It is therefore, possible to offer the designer alternatives with the lowest risk of propagation of obsolescence; i.e. to define the propagation absorbers, amplifiers, and carriers entities.

4. Illustrative case and SE-CEM methodology implementation

EOLE is a balloon probe system to provide meteorological data to users. It has two subsystems: an acquisition subsystem "in the air" and a ground processing subsystem. The whole architecture is defined in [13].

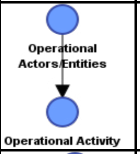
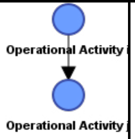
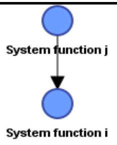
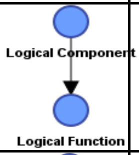
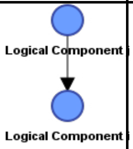
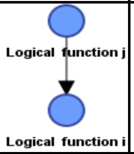
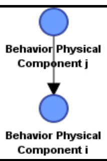
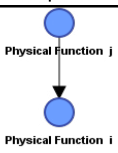
4.1. *The analysis of changes to the EOLE system*

System modeling. We redo the modelling from the external analysis (operational analysis) to the internal modeling (EPBS) using Capella. The second column of Table.3 lists the most important diagrams for each level of the ARCADIA methodology for this purpose. For example, the model of the physical architecture in the fifth row, shows the links between different functions and component within their own sub-systems.

Dependency modeling. Based on the diagrams of the first step, the matrices of dependencies are created (Design Structure Matrix (DSM) and Domain Mapping Matrices (DMM)) where a "1" indicates a dependency between the couple of entities of the architecture. However, the linkage data in these matrices are different from each other because at each ARCADIA methodology level, the system is modelled from a given perspective. In the first level, the matrix provides the links between the external actors / entities. It is a macro-representation of the environment of the system. It gives the DSM_{op} which links the external actors and the DMM_{op} describing the interactions through their functions.

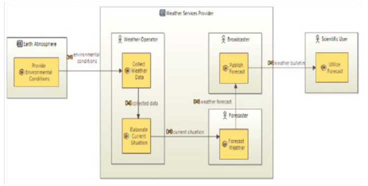
In the second step, the system and its functions are described. So we have the DSM_{sys} that connects the internal functions, and DMM_{sys} showings the system links with its environment. In the third level: logical architecture, we divide the system into subsystems. The DMM_{log} fixes "who does what" in the logical analysis of the system. For the physical architecture, the creation of the final architecture is done by defining the nodes physical components that contain behaviors physical components, i.e. DMM_{nodp}. DMM_{phi} models the physical functions of the components; "who does what inside the system". The last step is the association of Configuration Item with the different system components, DMM_{config}.

Bayesian networks. To create Bayesian networks, the ".csv" files (DSMs, DMMs) obtained from the previous step

ARCADIA levels	The entities of ARCADIS's levels	Rules	Bayesian network(BN)		Modalities for nodes / CPT(conditional probability table) for arc
			Structure (node/arc)	Structure description (ordinary, orphan, childless)	
1- Operational Analysis	Operational Actors/Entities	***R1.1*** : "Each operational actor or entity is represented by a node"		Ordinary node	-Activated / Deactivated ….
	Operational Activity	***R1.2*** : "Each operational activity is represented by a node"		Ordinary node	-Feasible/ Not Feasible -Obsolete /Not Obsolete, -Activated / Deactivated…
	Operational Actors/Entities and Operational Activity Exchange :	***R1.4***: " **If** an Operational Actors/Entities executed an Operational Activity then do ***R1.1*** for Actors/Entities and ***R1.2*** for Operational Activity and create an arc from the Operational Actors/Entities to the Operational Activity "		Function is a child nodes and Actor/Entity is a parent +unidirectional arc sense from Actor/Entity to Function	**Operational Activity**: **Feasible** | **Not Feasible** **Operational Actors/Entitles** – **Activated**: 100 % | 0 % **Deactivated**: 0 % | 100 %
	Operational Interaction	***R1.3***: **"If** there is correlation , dependence, or respectively, between/from an Operational Activity **i** and an Operational Activity **j, then** do ***R1.2*** for **i** and ***R1.2*** for **j** and create an arc from **i** to **j**."		**j** is a child node and **i** a parent node +unidirectional arc sense from **i** to **j**	**Operational Activity j**: **Feasible** | **Not Feasible** **Operational Activity i** – **Obsolete**: (1-X)% | X % **Not Obsolete**: Y % | (1-Y) % X and Y are real numbers
2-System Analysis	System Function	***R2.1:*** "Each system function is represented by a node"		Ordinary node	-Feasible/ not Feasible -Obsolete /Not obsolete, -Required /Not required…
	Functional System Exchange	***R2.2:*** " **If** there is correlation , dependence, or respectively, between/from a System function **i** and/to a System function **j, then** do ***R2.1*** for i and ***R2.1*** for **j** and create an arc from **i** to **j"**		**i** is a child node and **j** is a parent node +unidirectional arc sense from **j** to **i**	**System function j**: **Feasible** | **Not Feasible** **System function i** – **Feasible**: (1-X)% | X % **Not Feasible**: Y % | (1-Y) % X and Y are real numbers
3-Logical Architecture	Logical Component	***R3.1:*** "Each Logical Component is represented by a node"		Ordinary node	Obsolete /Not obsolete, -required /Not required, ….
	Logical Function	***R3.2:*** "Each system function is represented by a node"		Ordinary node	-Feasible/ not Feasible -Obsolete /Not obsolete, -Required /Not required…
	Logical Component and Logical Function Exchange	***R3.3:*** "**If** a Logical Component executed a Logical Function, then do ***R3.1*** for Logical Component and ***R3.2*** for Logical Function and create an arc from Logical Component to Logical Function"		-Logical Function is a child nodes and Logical Component is parent +unidirectional arc sense from Logical Component to Logical Function	**Logical Function**: **Feasible** | **Not Feasible** **Logical Component** – **Activated**: 100 % | 0 % **Deactivated**: 0 % | 100 %
	Logical Component Exchange	***R3.4:*** " **If** there is a relation between two logical components **i** and **j** , **then** do ***R3.1*** for A and R3.1 for B and Create an arc from **j** to **i"**		**i** is a child node and **j** is a parent node +unidirectional arc sense from **j** to **i**	**Logical Component j**: **Feasible** | **Not Feasible** **Logical Component i** – **Activated**: (1-X)% | X % **Deactivated**: Y % | (1-Y) % X and Y are real numbers
	Logical Function Exchange	***R3.5:*** " **If** there is correlation , dependence, or respectively, between/from a Logical function **i** and/to a Logical function **j, then** do ***R3.2*** for i and ***R3.2*** for **j** and create an arc from **i** to **j"**		**i** is a child node and **j** is a parent node +unidirectional arc sense from **j** to **i**	**Logical function j**: **Feasible** | **Not Feasible** **Logical function i** – **Feasible**: (1-X)% | X % **Not Feasible**: Y % | (1-Y) % X and Y are real numbers
4-Physical Architecture	Physical Component	***R4.1:*** "Each Physical Component is represented by a node		Ordinary node	-Activated / Deactivated - Obsolete /Not obsolete….
	Physical Function	***R4.3:*** "Each Physical Function is represented by a node"		Ordinary node	-Feasible/ Not Feasible -Obsolete /Not obsolete, -Required /Not required…
	Physical Component Exchange	R4.5: **"If** there is a relation between two Physical components **i** and **j** , **then** do do ***R4.1*** for **i** and ***R4.1*** for **j** and create an arc from **j** to **i**	 Behavior Physical Component j Behavior Physical Component i	**i** is a child node and **j** is a parent node +unidirectional arc sense from **j** to **i**	**Physical Component j**: **Activated** | **Deactivated** **Physical Component i** – **Activated**: (1-X)% | X % **Deactivated**: Y % | (1-Y) % X and Y are real numbers
	Physical Function Exchange	***R4.6*** "**If** there is correlation , dependence, or respectively, between/from a Physical function **i** and/to a Physical function **j, then** do ***R4.3*** for **i** and ***R4.3*** for **j** . Create an arc from **j** to **i"**		**i** is a child node and **j** is a parent node +unidirectional arc sense from **j** to **i**	**Physical function j**: **Feasible** | **Not Feasible** **Physical function i** – **Feasible**: (1-X)% | X % **Not Feasible**: Y % | (1-Y) % X and Y are real numbers

ARCADIA Levels | System Modelling | Dependency Models (MD) | Bayesian Networks (BNs)

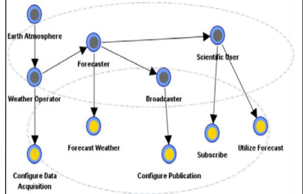
Operational Analysis



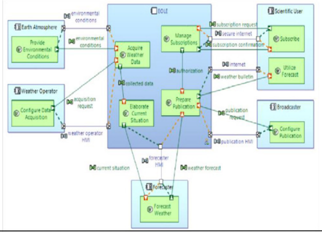
DSM op

DMM op

	Entities / Actors					Functions of Entities and Actors					
	Earth Atmosphere	Weather Operator	Scientific User	Broadcaster	Forecaster	Provide Environmental Conditions	Configure Data Acquisition	Subscribe	Utilize Forecast	Configure Publication	Forecast Weather
Earth Atmosphere						1					
Weather Operator	1						1				
Scientific User					1			1	1		
Broadcaster					1					1	
Forecaster		1									1



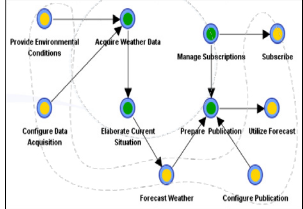
System Analysis



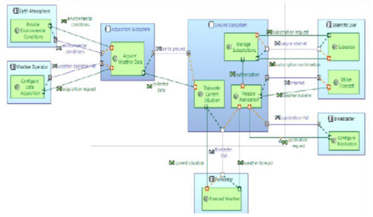
DSM sys

DMM sys

	System Functions				Functions of Entities and Actors					
	Acquire Weather Data	Manage Subscriptions	Prepare Publication	Elaborate Current Situation	Provide Environmental Conditions	Configure Data Acquisition	Subscribe	Utilize Forecast	Configure Publication	Forecast Weather
Acquire Weather Data				1	1	1				
Manage Subscriptions			1				1			
Prepare Publication								1	1	1
Elaborate Current Situation										1



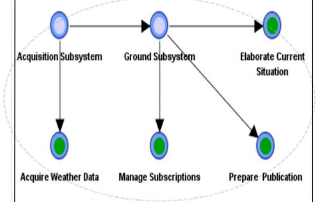
Logical Arcitecture



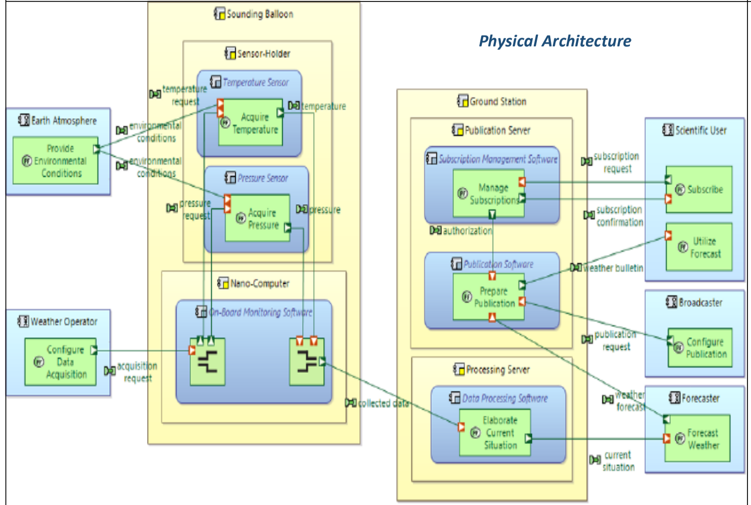
6

DMM log

		Logical Subsystems (ArchiLog)	
		Acquisition Subsystem	Ground Subsystem
System Functions	Acquire Weather Data	1	
	Manage Subscriptions		1
	Prepare Publication		1
	Elaborate Current Situation		1



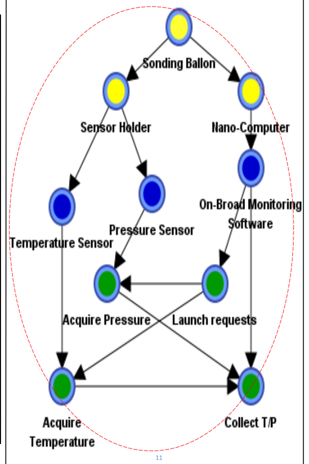
Physical Architecture



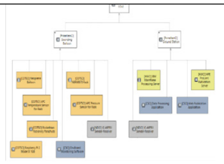
DMM nodp

DMM phi

		Node Physical Components						Physical Functions						
		Sensor-Holder	Nano-Computer	Publication Server	Processing Server	Sounding Ballon	Ground Station	Acquire Temperature	Acquire Pressure	Launch requests	Collect T/P	Manage Subscriptions	Prepare Publication	Elaborate Current Situation
Behavior Physical Components	Temperature Sensor	1				1		1						
	Pressure Sensor	1				1			1					
	On-Broad Monitoring Software		1			1				1	1			
	Subscription Management Software			1			1					1		
	Publication Software			1			1						1	
	Data Processing Software				1		1							1



EPBS



DMM config

		Configuration Item					
		System CI	Prime Item CI	CSCI	HWCI	NDI	COTS
Node Physical Components	Sensor-Holder						1
	Nano-Computer						1
	Publication Server				1		
	Processing Server				1		
	Sounding Ballon		1				1
	Ground Station		1				
Behavior Physical	Temperature Sensor						1
	Pressure Sensor						1
	On-Broad Monitoring Software			1			
	Subscription Management Software			1			
	Publication Software			1			
	Data Processing Software			1			

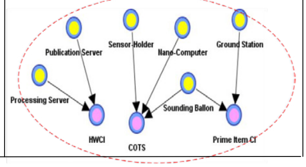


Table 3. From dependency Model to Bayesian network translation

are used. They allow to create systematically a Bayesian networks for each level of ARCADIA, cf. Table.3. Let us now take one example of these Bayesian networks, that of the fourth level. We take a sub-matrix of the large matrix (red frame) which shows the entities of the functional chain of ”acquisition of meteorological data”. In this functional chain, there are two nodes physical components (Sounding Balloon, Nano-Computer) and three behaviors physical components (Temperature Sensor, Pressure Sensor and On-Broad Monitoring Software). The modularities of the components are ”Obsolete” or ”Not Obsolete”. There are four functions (Acquire Temperature, Acquire Pressure, Launch Requests, Collect T/P) whose modularities are ”Feasible” or ”Not Feasible”. Once, the Bayesian network established, the modalities of the node identified and also their marginal and conditional probabilities, it becomes possible to make a set of experiments to understand the possible behaviour of the system based on various scenarios. A simple scenario is made as a set of ”What-If”. The idea is to know what happens if an event has occurred. The events associated to these scenarios are called evidences. There are *Hard* and *Soft* evidences. A hard evidence is an observed evidence. This means that the probability of the associated modality is put to 100% because there is no more uncertainty about its occurrence. This is the case for instance for an evidence concerning the fact that the ”Temperature Sensor” is ”obsolete”. The soft evidences are more nuanced because the observation gives a newly imposed probability to the modalities; the computation is then made for the whole Bayesian network. A soft evidence would be for instance $P(TemperatureSensor = Obsolete) = 70\%$ in the coming two years. Finally, using these two sets of evidences, basically three kinds of experiments can be performed using a Bayesian network: (i) prognostic (what if a parent node's probabilities are modified), (ii) diagnosis (what if a chid's probabilities are modified), (iii) robustness testing (through a set of scenarios to be able to find out the most interesting node that stops obsolescence propagation throughout the network).

Hereafter, we are going to set up 3 prognosis experiments and a diagnosis experiment in the BN of Physical Architecture. They allow giving recommendations for designers.

Experiments and interpretations. To run simulations, the Bayesian network of the fourth step of ARCADIA is chosen: the physical architecture. Indeed, this step gives a final idea of the physical composition of the system. We take the Bayesian network (RB) of the first functional chain of the metrological data acquisition , figure 4. The final function of this chain is to collect the pressure and temperature data. Our work consists in studying the impact of obsolescence of the 3 parent nodes on the realization of the function' collect' T/P'. So we generate 3 obsolescence scenarios of 3 physical components: 'Temperature sensor', 'Pressure sensor' and 'On-Board Monitoring Software'. Figure 4 also shows two indicators of the costs of obsolescence remediations, represented by diamonds(Mitigation cost 1 , Mitigation cost 2). Indeed, if obsolescence affects one of three components, the costs of implementing solutions are associated to have a system that resists despite the occurrence of this obsolescence. For example, the cost of remedying the

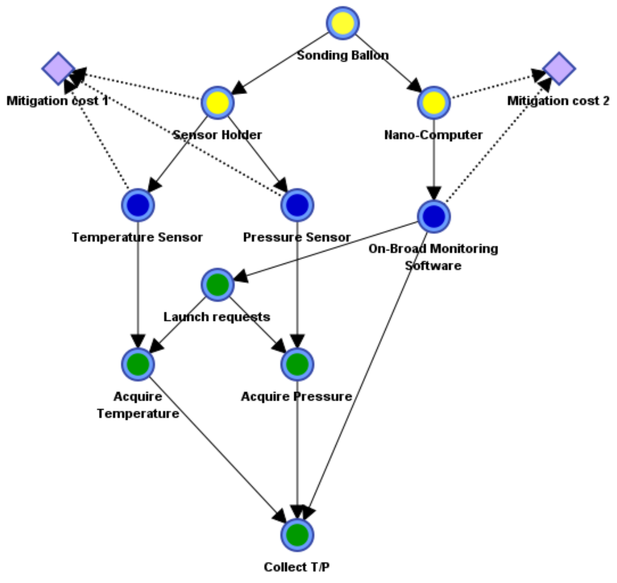


Fig. 4. BN of data acquisition functional chain

obsolescence of the temperature sensor is 20000 euro (7th row of table 4). Before starting the simulations, we must fill in the Conditional Probabilities Tables of Bayesian networks (CPTs): tables 5, 6 and 7 correspond to the CPTs of the functions of the 3 physical components that we will apply the obsolescence scenarios.

For 'acquire temperature' function (table5), the rate of realization depends on the realization of the 'launch request' function and on the obsolescence of the 'Temperature Sensor'. If the temperature sensor is not obsolete, we have a rate of 99.990% (second row) knowing that the 'launch request' function is feasible. In the case of obsolescence of this sensor, the analysis is based on the performance of the remediation chosen to maintain the resilience of the system. It is assumed that the remediation of this obsolescence gives a rate of feasibility to the function almost close to the old temperature sensor: 99.985% (first row). The realization rate is almost null if 'launch request' is not feasible

For the 'Acquire Pressure' function, it is assumed that the remediation of the obsolescence of the pressure sensor is less performing. So here we have a function realization rate equal to 99.990% if the sensor is not obsolete and 99.900% if a remediation is used, see table 6. The ”launch request” function depends only on On-Broad Monitoring Software obsolescence (table7) . New software technologies are more efficient than old ones in most cases. So table 7 is filled based on this idea : the performance rate of the function is higher when the software is obsolete (the remediation is used in this case) .

The simulation session for the prognosis is as follows. First, the model is based on the total absence of obsolescence. Figure 5.(a) shows that the performance of 'collect T/P ' function is 99.96%. When simulating the occurrence of temperature sensor obsolescence *(Scenario 1)* (Figure 5.(b)), we only notice a variation to the 3rd decimal of the performance (not perceptible). When the obsolescence of the pressure sensor *(Scenario*

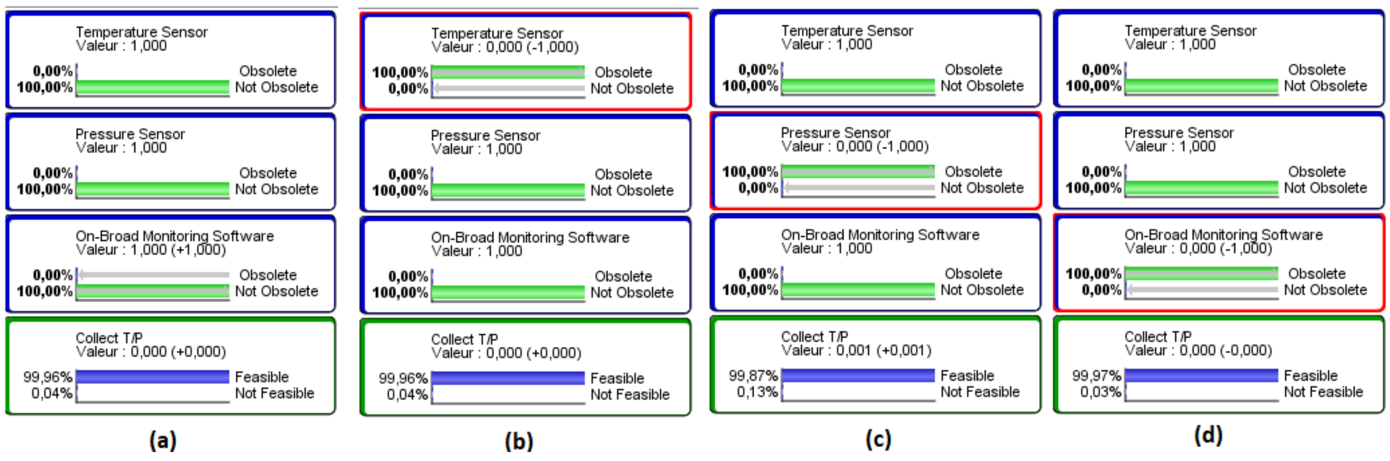


Fig. 5. (a) No obsolescence ; (b) Scenario 1: The obsolescence of temperature sensor ; (C) Scenario 2:The obsolescence of pressure sensor; (d) Scenario 3: The obsolescence of' On-Broad Monition Software' .

2) occurs, the variation is then more remarkable. From a performance of 99.96% to 99.87% (Figure 5.(c)). 'On-Broad Monition Software' obsolescence *(Scenario 3)* increases the rate of realization of 'collect T/P' function from 99.96% to 99.97% (Figure 5.(d))

We can assess the quality of the remediation and ultimately the resilience of the system designed for this remediation. In addition, each of these scenarios is associated with an overall cost shown by the cost indicators.

Sensor Holder	Pressure Sensor	Temperature Sensor	Cost (euro)
Obsolete	Obsolete	Obsolete	50000
		Not Obsolete	50000
	Not Obsolete	Obsolete	50000
		Not Obsolete	50000
Not obsolete	Obsolete	Obsolete	40000
		Not Obsolete	20000
	Not Obsolete	Obsolete	20000
		Not Obsolete	0

Table 4. Cost of remediations

Launch requests	Temperature Sensor	Acquire Temperature	
		Feasible	Not Feasible
Feasible	Obsolete	99.985	0.015
	Not Obsolete	99.990	0.010
Not Feasible	Obsolete	0.001	99.999
	Not Obsolete	0.001	99.999

Table 5. CPT of ' Acquire Temperature'

Launch requests	Pressure Sensor	Acquire Pressure	
		Feasible	Not Feasible
Feasible	Obsolete	99.900	0.100
	Not Obsolete	99.990	0.010
Not Feasible	Obsolete	0.001	99.999
	Not Obsolete	0.001	99.999

Table 6. CPT of ' Acquire Pressure'

Finally , The simulation session for the diagnosis is as follows. To do this, we fix a hard evidence that the final function' Collect T/P' is 100% not Feasible and we deduce the obsolescence probabilities of the nodes that cause this malfunction (the critical nodes). For example, in the graph in figure 4, we want to know which component is the most critical. In other words, the component that contributes the most to this dysfunction. Figure 6 shows this simulation. So, the most critical component is 'On-Broad Monition Software' because it has the highest probability of obsolescence (49.83%).

On-Broad Monitoring Software	Launch requests	
	Feasible	Not Feasible
Obsolete	99.995	0.005
Not Obsolete	99.990	0.010

Table 7. CPT of ' Launch requests'

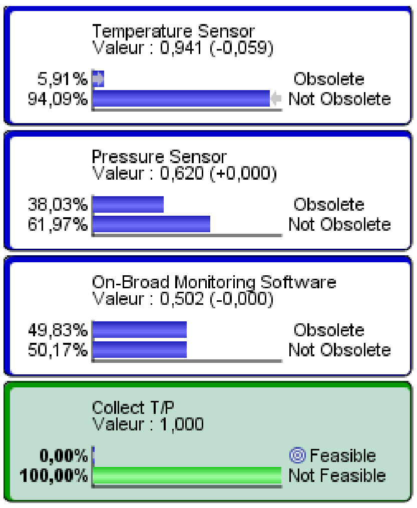


Fig. 6. Diagnosis Simulation

4.2. Results Discussions

In simple systems such as the partial EOLE subsystem shown in Figure 4, it is not complicated to calculate the probability of performing the 'Collect T/P' function, if one of the system entities (component, function, functionality) is affected by obsolescence. In contrast, when the system has a large number of components such as space or aeronautic system, it is quite difficult to determine all the possible combinations of obsolescence probabilities of the different entities in the system. This work made it possible to bring back the combinations of all possible cases to a graphical calculation of conditional probability over the lifetime of the system. This is done using the Bayes' theorem and the acyclic graph structure of the Bayesian network. If we change a probability, a propagation occurs. For example, the probability of achieving the overall functionality of the system in the presence of this propagation can be deduced. The contribution of this work also consists in knowing the combination of obsolescence probability that generates the highest rate of system dysfunction. In addition, the critical entity can be determined. Indeed, if we assume that the system has such a probability of non-realization, we can deduct from the graph the probabilities of obsolescence from each entity.

5. Conclusion

Component and system changes are increasing rapidly to keep pace with technological change. However, this creates obsolescence to old components and affects the system architecture. In this work, a structured methodology was proposed that contains several tools. System engineering tools were initially used to describe the system by applying the ARCADIA methodology and Capella software. Then, DSM and DMM matrices were used, which are tools of the engineering change. In the end, to study the impact of change and the impact of obsolescence in particular, we use the BayesiaLab tool, which is based on Bayesian networks. This method is applicable to the complex system, because it is based on probabilistic graphs. In addition, it can help designers to study the impact of changes on product architecture and to make prognostic and diagnosis scenarios simulations to assess the risks and costs of each design choice of a system to make it more resilient to obsolescence.

Acknowledgement

This research work is fully supported by the NExTRegio project of IRT Railenium. The authors would like to thank SNCF Réseau for its support.

References

[1] Aguirre-Ollinger, G., F. Stahovich, T., 2001. A constraint-based tool for managing design changes. RedesignIT .

[2] BayesiaLab, 2018. BayesiaLab User Guide. https://library.bayesia.com/.

[3] Bradley, M., Dawson, R., 1998. An analysis of obsolescence risk in it systems. Software Quality Journal 7, 123–130. URL: https://doi.org/10.1023/A:1008808708860, doi:10.1023/A:1008808708860.

[4] Clarkson, P., Simons, C., Eckert, C., 2004. Predicting change propagation in complex design. Journal of Mechanical Design 126, 788–797. URL: http://oro.open.ac.uk/18626/.

[5] Cohen, T., Navathe, S., Fulton, R., 2000. C-far, change favorable representation. Computer-Aided Design 32, 321 – 338. URL: http://www.sciencedirect.com/science/article/pii/S0010448500000154, doi:https://doi.org/10.1016/S0010-4485(00)00015-4.

[6] Danilovic, M., Browning, T.R., 2007. Managing complex product development projects with design structure matrices and domain mapping matrices. International Journal of Project Management 25, 300 – 314. URL: http://www.sciencedirect.com/science/article/pii/S0263786306001645, doi:https://doi.org/10.1016/j.ijproman.2006.11.003.

[7] Fowler, H.W., 1995. The concise Oxford dictionary of current English. Clarendon Press Oxford University Press, Oxford New York.

[8] Gorbea, C., Spielmannleitner, T., Lindemann, U., Fricke, E., 2008. Analysis of hybrid vehicle architectures using multiple domain matrices, in: Kreimeyer, M., Lindemann, U., Danilovic, M. (Eds.), DSM 2008: Proceedings of the 10th International DSM Conference, Stockholm, Sweden, 11.-12.11.2008, pp. 375–387. URL: https://www.designsociety.org/publication/27453/.

[9] Holley, V., Yannou, B., Jancovic, M., 2011. Using the ff-dmm matrix to represent functional flow in product architecture, in: DSM 2011: Proceedings of the 13th International DSM Conference.

[10] Huang, G., Mak, K., 1999. Current practices of engineering change management in uk manufacturing industries. International Journal of Operations & Production Management 19, 21–37. URL: https://doi.org/10.1108/01443579910244205, doi:10.1108/01443579910244205, arXiv:https://doi.org/10.1108/01443579910244205.

[11] Jarratt, T.e.a., 2005. Engineering change, in: Clarkson, J., Eckert, C.E. (Eds.), Design process improvement : a review of current practice. Springer, London U.K. chapter 10, pp. 262–285. URL: https://www.springer.com/us/book/9781852337018.

[12] Roques, P., 2015. MBSE with the ARCADIA Method and the Capella Tool.

[13] Roques, P., 2017. Systems Architecture Modeling with the Arcadia Method: A Practical Guide to Capella. Elsevier Science. URL: https://books.google.fr/books?id=Qj3jCwAAQBAJ.

[14] Sandborn, P., 2013. Design for obsolescence risk management. Procedia CIRP 11, 15 – 22. URL: http://www.sciencedirect.com/science/article/pii/S2212827113005477, doi:https://doi.org/10.1016/j.procir.2013.07.073. 2nd International Through-life Engineering Services Conference.

[15] Sebokwiki, 2019. Guide to the Systems Engineering Body of Knowledge (SEBoK). https://www.sebokwiki.org/wiki/Guide_to_the_Systems_Engineering_Body_of_Knowledge_(SEBoK).

[16] Solomon, R., Sandborn, P.A., Pecht, M.G., 2000. Electronic part life cycle concepts and obsolescence forecasting. IEEE Transactions on Components and Packaging Technologies 23, 707–717. doi:10.1109/6144.888857.

[17] TechnicalReport, . Processes for engineering a system. URL: https://doi.org/10.4271/eia632, doi:10.4271/eia632.

[18] Ullah, I., Tang, D., Yin, L., 2016. Engineering product and process design changes: A literature overview. Procedia CIRP 56, 25 – 33. URL: http://www.sciencedirect.com/science/article/pii/S2212827116310022, doi:https://doi.org/10.1016/j.procir.2016.10.010. the 9th International Conference on Digital Enterprise Technology Intelligent Manufacturing in the Knowledge Economy Era.