Introduction Safety Requirements in Component of an Embedded Software for Self-defense Against the Failures with a Technique Combining Concept Lattice and Graph.

Lala Madiha Hakik
Faculty of Sciences and Techniques,
University Hassan I, BP 577. Settat, Morocco,

Rachid El Harti
Faculty of Sciences and Techniques,
University Hassan I, BP 577. Settat, Morocco,

Abstract

AUTOSAR applications are decomposed into software components that interact with each other. They can exchange either data or services, via interfaces[1].
They cannot directly call the services of basic software, that is to say the scheduler, the communication bus, equipment, etc.
In this paper we worked on software architecture AUTOSAR, object of the thesis that Caroline Lu[1] *found failings at components especially functionality part of modules.*
To remedy it has adopted a technique to develop software defense such a configurable external component, based on the observability and controllability of the mechanisms provided by standard software architecture Automobile emerging AUTOSAR[2].
Our approach is remodularization at the component by introduction requirements for modification of the functionality at the module using the Galois lattice with Formal concept analysis FCA and directed labeled graph for Self-defense Against the Failures.

Keyword: *Embedded Software Automobile, , Component, Module, Remodularization, FCA, Graph, Self-defense Against the Failures*

1. Introduction

The embedded software in a computer is part of the mechatronic system vehicle. Thus, hardware faults to electronics and the environment (electromagnetic interference, temperature variations...etc) are sources of errors that can cause failure of the software.
They can cause, for example, by corruption of data, parameters, even of code segments.
Moreover, the complexity of software is a factor in increasing the number of software faults remaining, these faults are likely to appear throughout the process of software development: when specifications of developments design, of implementation manual (possible error Interpretation) or automatic.
In automobile, the concept of safety of operation is characterized by availability property therefore fitness of the use of a vehicle, it relates also reliability whence capacity to ensure continuity of service and also maintainability which is an ability to maintain in operating condition. Finally, the notion of security which is a system suitability did not know catastrophic event.
In this context the thesis work of Caroline Lu [1] has ensured the robustness of embedded software AUTOSAR by adding a component defense contributing to improve safety of operation of system.
In our approach we worked on architecture fault-tolerant, for software platforms of modular Type and multilayer the same than used by Caroline Lu [1] focusing on the same requirements and the same failures.
In our case, we conducted a remodularization at the component level by introduction of Safety requirements for modification of the functionality at the component level using the Galois lattice with Formal concept analysis FCA and directed labeled graph for Self-defense Against the Failures.
This method combining concept lattice with FCA and directed labeled graph was approved by a formalism.
we recall our approach comes after identification of errors by the existing failure detector of embedded system.
In this paper, section 2 presents our example, then we describe the approach in Section 3. Related work is presented in Section 4 and then we conclude in Section 5.

2. Illustration

This section presents the system studied [1] is Embedded on a microcontroller 16 bits, S12XEP100

Freescale. It has the particularity to include mechanisms of memory protection hardware (MPU).
The software architecture of our case is of type Autosr (extracted from the thesis of Caroline Lu Toulouse University [1] (see figure.1)) superimposes 4 layers. The basic software has two layers of abstraction: AUTOSAR Service Layer and AUTOSAR MicroController Abstraction Layer MCAL.
Abstraction layer of microcontroller contains only Module General Purpose Timer (GPT) which manages hardware timers. The service layer is reduced to a real-time executive, called "Trampoline OS" [Bechennec et al. 2006]. This software is open source, developed by Irccyn, from the OSEK OS and AUTOSAR OS specification. The communication layer AUTOSAR RTE is generated automatically in configuration information from the application layer and basic software. This code generation was carried out using the commercial tool DaVinci 2.2 MICROSAR RTE TM Vector (see figure 1).
The application layer includes 4 software components with interfaces AUTOSAR. The "air conditioning" component consists of the adaptation and porting an existing automotive module. The "airbag" components and " torque transmission "are synthetic. The rest being represented by timers. The last synthetic component "stub" represents the rest of the environment application. It sends to the three other components and data from the sensors and other computers, they need (see figure 1).
The coexistence of air conditioning, airbag and torque transmission functions at the same computer is only illustrative and may be unrealistic today.

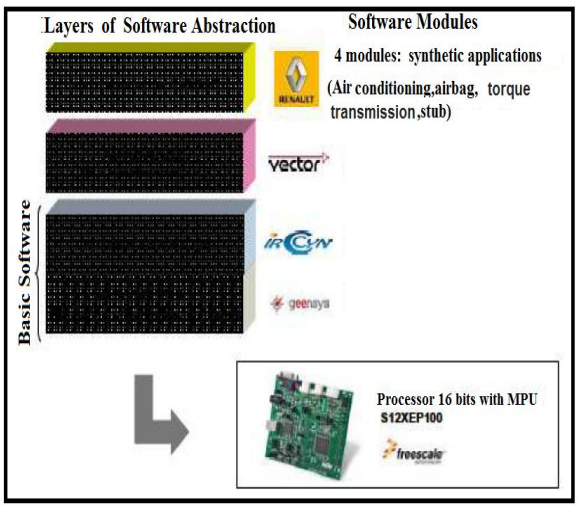

Figure 1. Multi-level Software Architecture AUTOSAR extracted from the thesis of Caroline Lu Toulouse University [1].

We are interested in the failures in components including functionality part of modules.
To remedy this problem, Caroline Lu [1] has found failings at the component level especially functionality part of modules has developed a software defense such a configurable external component, based on the observability and controllability of the mechanisms provided by standard software architecture automobile emerging AUTOSAR.
Our work has focused on the functional patterns of 3 components: air conditioning, airbag, torque transmission, while using the security requirements for optimization of the component without errors.
In this paper we have limited ourselves in one case of the functional pattern air conditioning component (see figure2 composed of 3 modules: a Manual Control, a filtering and an operating mode with data consumed and produced data), the same methodology is applicable to other components.

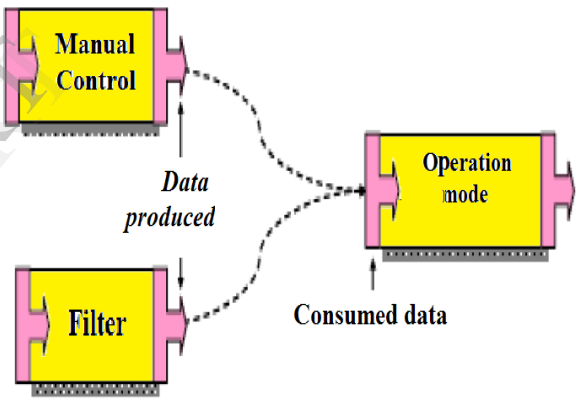

Figure 2. Functional pattern of "air conditioning" modules extracted from the thesis of Caroline Lu Toulouse University [1].

We believe that the Formal Concept Analysis (FCA) and graph can bring interesting ways to solve this problem because this technical method led us to a formalism resulting in a functional pattern for each component allowing it of appropriating; with safety requirements, self-defense against the alleged failures.
In our approach, we focused only on safety requirement (table 1), for preparing the concept lattice of figure 3.

Table1. Safety requirement and type of specific failure[1].

Safety requirement	Type of specific failure
R1: The calculation of operation mode should only be done when all of the input from the manual control data and the filter are available.	F1:Dataflow : Value exchanged unwanted F2: Dataflow: Execution sequence unwanted

3. Proposed approach

The Formal Concept Analysis (FCA) [3] [4] [5] [6] [7] [8] is a technical data analysis that allows you to group entities with common characteristics. A concept is a maximal set of entities (extension of the concept) sharing a maximal set of characteristics (intension of the concept). The FCA is used in software engineering for solving several problems [4].

Configurations In the context of our problem, we studied one configuration with FCA.

The configuration with FCA is to define a formal context C: the set O of entities studied (or formal objects) Set A of characteristics (or formal attributes) and the relationship $R \subseteq O \times A$.

The formal context associated to the functional pattern of modules "Air conditioning" component in Figure 2. This context is represented by the triplet (O, A, R).

Context (formal context C).

- O is the set of the modules with independent requirements.
- A is the set of independent requirements or gathered by the relationship "And", whether data consumed or produced data.
- R is the relation between objects and attributes, it is a relationship of safety and control data " Provide" or "Available ".

Table2. Legend of figure 3.

Objects	Attributs
CM: Control Module Manual **F**: Filtering Module **MF**: Operation mode **E1**: The input data from the manual control are available. **E2**: The input data from the filter are available.	**DC**: Data consumed **DP**: Data produced **E1**: The input data from the manual control are available. **E2**: The input data from the filter are available. **E1 And E2**: The input data from the manual control and filtering are available.

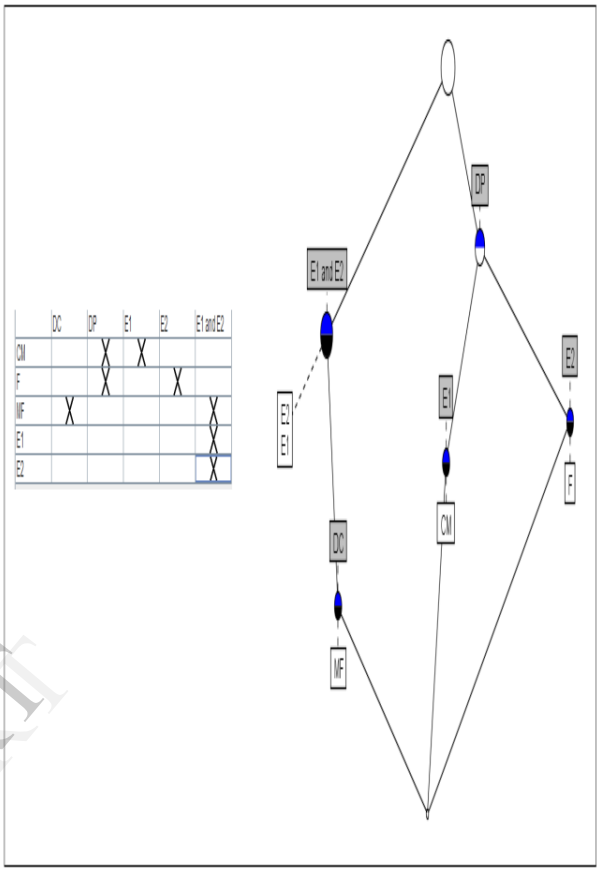

Figure 3. Formal context 1 and lattice T(C1)
- Functional pattern of "air conditioning" -

3.1. Formalization of result of the obtained lattice

The lattice of Figure 3 is used as follows:

- For all concepts [{CM,F},{DP}], [{CM},{DP,E1}], [{F},{DP,E2}], [{MF},{DC,E1 And E2}], objects and attributes are considered as nodes characterized by: **CM, F, MF, DP, DC, E1, E2, E1 And E2.**

- The relationship between objects and attributes are represented by edges connecting each pair of nodes as an example for the concept [{CM}, {DP, E1}] where the nodes E1 and CM are connected by the edge (E1, CM) image of couple (**Attribute**, **Object**).

It is found that all the conditions are met to define a graph oriented, object of Figure 4 below from the result of the lattice of the figure 3.

Definition 1 (Oriented Graph) [9]:
A graph G is a mathematical structure defined by a pair (N, E) where N is a set of objects called nodes or vertices and E part of N * N which represents a set of

arcs (also called edges) each connecting a pair of nodes.
This general definition is a directed graph distinguishes two vertices s1 and s2 the edge (s1, s2) of the edge (s2, s1).

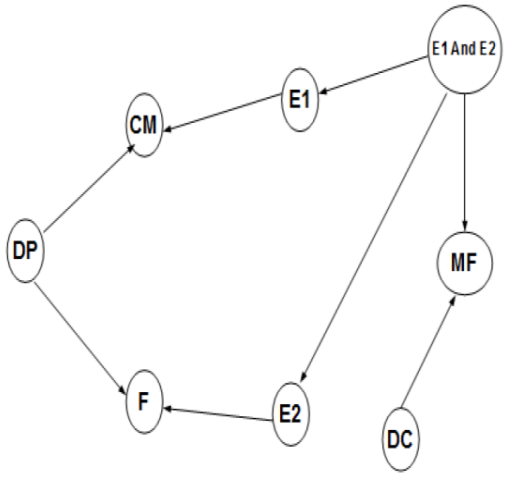

Figure 4. Oriented Graph result of the lattice of figure 3.

to permit reading a functional pattern of the modules "Air Conditioning", we did call the labeled directed graph because it exists in the lattice relations "Provide" or "Available", object of Figure 5 applied to the following way:
- The labeled them "provide" is used between two nodes of the same type whether of modules or requirements.
- The labeled "Available" is used between two nodes of different types.

Our approach to labeling is inspired by part of the thesis Adil Anwar, Toulouse University [9] , treating Directed Labeled Graph.

Definition 2 (Directed Labeled Graph) [9]:
Labeling of Graph G is a function l, or partial defined N ∪ E to a set of labels L (l: N ∪ E → L). For every element x in the field, the element l(x) is called the label of x.
The three types most common for labeling graphs are:
• The total labeling: in this case is the total function (defined on a set N ∪ E).
• The labeling of node: the domain of definition of l is N.
• The arc labeling: the domain of definition of l is E.
Typically, L is a set of integers but can also be a set of strings.
A labeled graph G is thus fully defined by the triplet (N, E, L) where N is an set of nodes, a set of edges E and l a function defined on labeling N ∪ E.

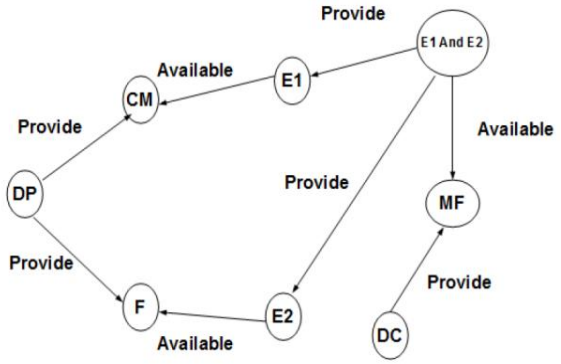

Figure 5. Directed Labeled Graph result of the lattice of figure 3.

We find that our approach, led us so far to define functional pattern model (figure 5) of the component "Air conditioning " and will apply to other components in observing the following definition:

Definition 3 (functional pattern Model "FPM"):
We define a functional model " FPM " as a directed labeled graph defined by the triplet (N, E, l)
(FPM = (N, E, l)) where N is the set of elements of model to represent (modules, independent requirements, requirements gathered, data produced, consumed data) .
-E is a set of edges representing the relationships between elements of N (E ⊂ N * N).
An arc is thus uniquely defined by the source node and node destination.
- L is a function that allows to describe the nodes and arcs of the graph. In reality the allows qualify the type of nodes (module independent requirement, requirement gathered, data produced, consumed data) and semantic relationships between these elements (Provide, available).
- The labeled "provide" is used between two nodes of the same type whether of modules or requirements.
- The labeled "Available" is used between two nodes of different types.
- l is defined by: l: N ∪ E→L with L is the set of possible labels in the model.

4. Related Work

The automobile approaches of software faults is still relatively little thorough, which is explain by the recent development of embedded computing. In Renault, for example, failure analysis for computers are guided by a model of physical faults. The equivalent in the software does not yet exist[1].

In the European EASIS project, software faults are mixed with hard faults [1] [11]. software faults are described: the scheduling mistakes, errors of communication between software components, and functional faults[1].

In the AUTOSAR consortium, each working group expresses its assumptions fault for the software module which is supported [1] [10].

The method of Caroline Lu[1] is to adopt a technique to develop software defense such a configurable external component, based on the observability and controllability of the mechanisms provided by standard software architecture Automobile emerging AUTOSAR[2].

Different automated approaches have been proposed to restructure object systems. We cite three: the clustering algorithms, algorithms based on meta -heuristics and those based on the FCA. The first aim to restructure system by the distribution of some elements (eg classes, methods , attributes) in groups such that the elements of a group are more similar to each other with elements of other groups [13] [14] . Approaches to restructuring based on meta-heuristic algorithms [15] [16] are generally iterative stochastic algorithms, progressing towards a global optimum of a function by evaluating a certain objective function (eg characteristics or quality metrics). Finally, the approaches based on FCA [17] [18] provide an algebraic derivation of hierarchies of abstractions from all entities of a system. Reference [4] presents a general approach for the application of the FCA in the field of object-oriented software reengineering. Recently, we added the dimension of exploration using the FCA [7] [8] and we have extended our research to introduce a technique of adding a new functionality in a package with FCA[12].

Our approach is remodularization at the component by introduction requirements for modification of the functionality at the module using the Galois lattice with Formal concept analysis FCA and directed labeled graph for Self-defense Against the Failures.

5. Conclusion and discussion

In this article, we present and illustrate a theoretical case and propose a technique of introduction requirements for modification of the functionality at the module using the Galois lattice with Formal concept analysis FCA and directed labeled graph for Self-defense Against the Failures.

This method is approved by a formalism for the elaboration of a functional pattern model.

6. References

[1] C. Lu. Robustesse du logiciel embarque multicouche une approche reflexive application a l'automobile. Thesis. Toulouse *Univ*ersity. *2009.*

[2] Automotive Open Standard Architecture, http://www.autosar.org

[3] B. Ganter and R. Wille. Formal Concept Analysis. Mathematical Fondations. Spinge. 1999.

[4] T. Tilley, R. Cole, P. Becker, P.W. Eklund. A survey of formal concept analysis support for software engineering activities. In Int. Conf. Fomal Concept Analysis (ICFCA 2005), pages 250-271, 2005

[5] G. Arévalo, S. Ducass, and O. Nierstrasz. Lessons leaned in appling fomal concept analysis to reverse engineering. In Proceeding of the Third international conference on Fomal Concept Analysis, ICFCA'05, pages 95-112, Berlin. Heidelberg, 2005. Spinge-Velag.

[6] B. Ganter and R. Wille. Formal Concept Analysis. Mathematical Fondations. Spinge. 1999.

[7] Lala Madiha Hakik, Rachid El Harti . " *Exploring the Redistribution Classes of a Package with an Approach Based on Formal Concept Analysis* ", Vol.2 - Issue 12 (December - 2013), International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, www.ijert.org

[8] Lala Madiha Hakik, Marianne Huchard, Rachid El Harti et Abdelhak Djamel Seriai. Exploration de la redistribution des classes d'un package par des techniques d'Analyse Formelle de Concepts. The first conference in software ngineering (CIEL 2012), France, 2012.

[9] A. Anwar. Formalisation par une approche IDM de la composition de modeles dans le profil V*UML*. Thesis. Toulouse *Univ*ersity. *2009.*

[10] AM Salkham, Fault Detection, Isolatation and Recovery (FDIR) On-Board Software Master's Thesis, Chalmers *Univ*ersity *of technology, Gotebor, Sweden,2005.*

[11] J. Bohm, M. Menzel, X. Chen , J.M. Dressler, T.E*ymann, M. Hilte*r, T. Kimmeskamp, V. Quenda. Description of Fault T*ypes fo*r EASIS V2.0. Technical report, Jun 2005.

[12] Lala Madiha Hakik, Rachid El Harti . " Technique of Adding A New Functionality in A Package with An Approach Based on Formal Concept Analysis ", Vol.2 - Issue 12 (December - 2013), International Journal of Engineering Research & Technology (IJERT) , ISSN: 2278-0181 , www.ijert.org

[13] F.B. Abreu, G. Pereira, and P. Sousa. A coupling-guided cluster analysis approach to reengineer the modularity of object-oriented systems. In Proceeding of the confeence on Software Maintenance and Reengineering. CSMR 'OO, pages 13-, Washington, DC, USA, 2000. IEEE Compter Society Press.

[14] M. Bauer and M. Trifu. Architecture-aware adaptive clustering of oo s ystems. In Poceedings of the Eighth Euromicro Working Conference on Software Maintenance and Reengineering (CSMR 'O4), CSMR 'O4, pages 3-, Washington, DC, USA, 2004. IEEE Compter Society.

[15] M.O'Keeffe and M. i Cinneide. Seach-based refactoring fo software maintenance. J. Syst. Softw., 81(4): 502-216, April 2008.

[16] O. Seng, J. Stammel and D. Burkhart. Search- based determination of refactorings for improving the class structure of object-oriented systems, In Mike Cattolico, edito. GECCO, pages 1909-1916. ACM, 2006.

[17] G.Snelting. Software reengineering based on concept lattices. In CSMR, pages 3-10, 2000.

[18] P. Tonella.Concept analysis for module restructuring. IEEE Trans. Software Eng..27 (4): 351-363, 2001.