Redundancy Allocation in Automotive Systems using Multi-objective Optimisation

Indika Meedeniya[1], Aldeida Aleti[1], and Barbora Buhnova[2]

[1]Faculty of ICT, Swinburne University of Technology
Hawthorn, VIC 3122, Australia
imeedeniya,aaleti@swin.edu.au
[2]Faculty of Informatics, Masaryk University,
60200 Brno, Czech Republic
buhnova@fi.muni.cz

Abstract. Redundancy allocation is a widely used method for the reliability improvement of complex embedded systems in the automotive domain. The allocation of redundant components can affect other non-functional quality attributes of the system, which are as important as reliability and very often conflicting with each other, such as cost and response time. Consequently, to find the good system design options, multi-objective optimisation methods can be employed which find a trade-off between these conflicting attributes. While cost and reliability have been already studied in the context of an optimisation problem, the impact of redundancy allocation to the response time has not yet been considered. The approach taken in this paper, considers reliability, cost and response time as optimisation criteria and employs a multi-objective Ant Colony Optimisation algorithm to find the near optimal set of system designs. Our approach has been implemented in the ArcheOpterix framework and illustrated with a case study from the automotive industry.

Key words: Redundancy Allocation, Architecture Optimisation, Automotive Software, Response Time, ArcheOpterix

1 Introduction

The automotive industry has been significantly influenced by the development in electronics and software systems during the last few decades [1]. Legacy mechanical, electrical and manual systems are being replaced by embedded systems such as Electronic Fuel Injection (EFI), Anti-lock Breaking (ABS), Intelligent Parking Assistance, Airbag and Adaptive Cruise Control (ACC). Component based architectures have been investigated and proposed by many researchers [2,3,4,5] to address the design challenges in complexity, efficiency and cost pressure. In finding better architecture alternatives, non-functional quality attributes, such as safety, reliability, performance, maintainability are at least as important as the functional requirements [6].

Reliability is one of the crucial aspects that should be considered when designing embedded architectures of dependable, safety critical systems such as in

the automotive domain [3]. Apart from improving the reliability in the level of system components, redundancy allocation is one of the most popular system reliability improvement techniques [7,8,9,10]. Unfortunately, employing redundant components to increase the reliability of the system can have a negative influence in the other non-functional quality attributes. This problem has been addressed mostly in component based software architecture development, where the trade-offs between cost and reliability were investigated. Apart from the cost, when redundancy is employed in automotive systems, additional overheads incur into the response time. Response time is also a critical and important aspect of automotive systems, not only in safety critical sub-systems but also in user interaction sub-systems, such as the navigation system [3].

The response time concerned on its own has already been the focus of an extensive research in the embedded systems [11,12,13,14]. However, to the best of our knowledge, an approach to tackle the trade-off between reliability and response time has not yet been investigated. The approach in this paper finds a set of non-dominated [15] architecture solutions for component redundancy allocation considering reliability, response time and cost simultaneously, via an automated process based on a multi-objective optimisation algorithm. As an optimisation method, the Ant Colony Optimisation (ACO) has been employed. To illustrate our approach, a case-study from the automotive industry is taken. Since the sub-system components which are also known as COTS (Commercial Off-The-Shelf) components, are manufactured by external vendors, we consider them as black-boxes where internal structure and behaviour is not known. The evaluation models and optimisation algorithm are implemented into the ArcheOpterix [16], an architecture optimiser tool which is able to analyse an existing system specification in AADL(Architecture Analysis and Design Language). An automotive case study of an ABS and ACC integrated system will be described as the demonstration of our approach.

The paper is organised as follows: section 2 summarises the research contributions in redundancy allocation, reliability and response time evaluations for component based architectures together with the applicability of design space exploration techniques. The system model for our presentation is described in section 3. Section 4 presents the architecture quality evaluation models which we have used in computing reliability, response time and cost. We present the Ant Colony Algorithm for solving the redundancy allocation problem in section 5 and in section 6 we briefly describe the ArcheOpterix tool which has been used for the implementation. The case study and the results are presented in section 7. Finally, the conclusions and our perspective future work is presented at the end in section 8.

2 Related Work

There is a considerable amount of research available, which addresses the Redundancy Allocation Problem (RAP)[17] in the systems architecture design. Component redundancy allocation has been widely used as a reliability improvement

technique for dependable embedded systems [10]. Coit et al. [17] introduced an approach solving the redundancy allocation problem defined as the use of functionally similar (but not necessarily identical) components in a way that if one component fails, the redundant part performs required functionality without a system failure. They have visualised the problem as the minimisation of cost incurred for the redundancy allocation while satisfying a user defined system reliability level. In [17], Genetic Algorithms have been proposed for the optimisation of component redundancy allocation, and Neural networks techniques have also been integrated in [8]. Kulturel-Konak et al. [18] has also contributed on the redundancy allocation problem and has presented Tabu Search as the design space exploration strategy. The RAP has been adapted to Ant Colony Optimisation (ACO) by Liang et al. [7]. Significant similarity on all the approaches is the view on the RAP as cost minimisation problem while satisfying the predefined reliability criteria. Grunske [19] addressed RAP by integrating multi-objective optimisation of the reliability and weight.

Response time for real-time embedded systems has been the focus of many research groups in different aspects. Estimating Worst Case Execution Time (WCET) for individual functions in the system is one of the key work carried out in the context. Fredriksson et al. [12] worked on more realistic estimations of WCET using the system model. Some approaches of architecture optimisation [13,20] have used WCET as a design constraint. The Palladio component model [21], has significantly contributed in estimating the timing and scheduling behaviour together with their sensitivity to the parameters of component based systems. A number of path based, state based and Non-Homogeneous Poisson Process (NHPP) based estimations of response time for software systems can be seen in Sharma et al. [14]. The authors also presented an approach that can be used to evaluate a single metric of response time when the system is expressed as Discrete Time Markov Chain (DTMC). It is highlighted that the majority of the approaches above considers the response time and performance of the system as independent objectives, but not together with RAP. In this paper, we capture the impact on response time of the system functionality in solving cost-reliability trade-offs in RAP. In evaluation of response time for each redundancy allocation alternative, we extend the work of Sharma et al. [14] by calculating the overhead of redundancy to connected components. Instead of considering reliability as a constraint, we provide the (near)optimal set of solutions having best trade-offs in reliability, response time and cost.

Numerous research contributions have been developed on design space exploration for system architecture design. In solving the RAP, most of the researches have used specific implementations for the presentation [7,8,13,17,22]. A number of generic approaches for design space exploration support has also been developed. Florontz et al. [23] developed a meta-model for architecture evaluation and optimisation. Modelling support for automotive embedded systems has also been the focus of many studies including [24,25]. In our approach, we take the advantages of flexibility and strength of our previous work ArcheOpterix [16]. The implementation supports system specification in AADL which is specifically

meant for embedded system architectures, resulting an enhanced applicability to the automotive domain. Having the in-built modelling, optimisation and specification support in ArcheOpterix, we extend the work on redundancy allocation problem as a multi-objective optimisation problem considering its impact on response time.

3 System Model

The approach presented in this paper is focused on component-based systems in the automotive domain. In this context, the term *Component* represents a system element, which in our case is a special purpose Electronic Control Unit (ECU). An ECU is a self-contained computer along with the software programmed in it, and is dedicated to fulfil a specific functionality, such as getting input from sensors and calculating the distance to the next vehicle. As the ECUs need to communicate with each other during the operation, they are connected via buses.

Inter-component communication is modelled as an execution transfer from one component to another. Most of the redundancy allocation work is focused on Series-Parallel (S-P) systems [7,17,18], including logical constructs for both serial and parallel execution. In the case of automotive systems, the models can be viewed as overlapped sets of S-P models (for individual system-level services), because the execution can start at different components (triggering the services). We employ parallel-redundant components model [7,17,18,19] used in S-P systems, where each component with its replicas connected in parallel is considered as a single unit called subsystem. Then the interaction among components (subsystems) can be formalised in terms of Discrete Time Markov Chains (DTMC) [26], where nodes correspond to subsystems (or equivalently components without replication) and transitions to execution passing among them. The effect of replication is then added not to the markov chain, but to the definitions of system-level properties (reliability, response time, cost).

The Figure 1 depicts the DTMC model of the system (left) and formation of subsystems when redundancy levels[1] are assigned for the components (right). The arcs are annotated with probabilities relevant to the transition, while certain nodes contain probabilities of execution initialisation. It should be noted that transition and execution initialisation probabilities are directly applied from the components to subsystems as the behaviour of the system is left unchanged with the redundancy allocation.

For the formal notation of the system, let $C = \{c_1, c_2, ..., c_n\}$, where $n \in \mathbb{N}$, denote the set of all components (before replication), and $\mathcal{I} = \{1, 2, ..., n\}$ denote the index set for components in C. An architecture alternative for the redundancy allocation problem in our approach is an assignment of redundancy levels for all components and is denoted as a. The set of all functions a is denoted as $A = \{a \mid a : C \rightarrow N\}$ which represents the set of all redundancy allocation candidates, where $N = \{n \mid 0 \leq n \leq nMax, n \in \mathbb{N}_0\}$ delimits the redundancy

[1] Number of additional components employed as redundancy

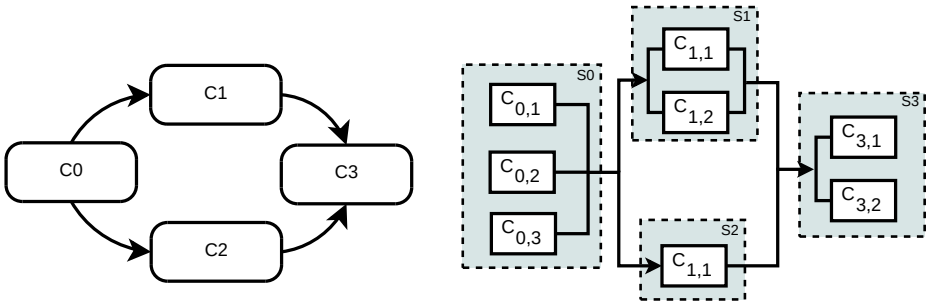

Fig. 1. System Model

level of a component. Note that, since C and N are finite, A is also finite. A component c_i together with its redundancies form a subsystem S_i, which can be uniquely determined by c_i, which we do along the formalization. Let the parameters of the system architecture be given as follows:

- Component Cost, $cst : C \to \mathbb{N}$, is the price associated with a single component; specified in ($)s.
- Failure Rate, $\lambda : C \to \mathbb{R}^+$, is the failure intensity of the exponential distribution of failure behaviour of a component [27]. Component failures in the model are assumed independent.
- Estimated Time per Visit, $et : C \to \mathbb{N}$, is the estimated time taken by component execution within a single visit of the component measured in milliseconds (ms).
- Redundancy Overhead, $\delta : C \times C \to \mathbb{N}$, is the estimated additional execution time for each of other components (in ms) due to a unit increment in redundancy level.
- Execution Initiation probability, $q_0 : C \to [0, 1]$, is the probability of initialising execution from that component. $\sum_{c \in C} q_0(c) = 1$
- Transfer Probability, $p : C \times C \to [0, 1]$, is the probability that execution transits component C_j after component C_i.

Additionally, we define two derived parameters, the first parameterized with a redundancy allocation function a.

- Sojourn Time per Visit, $st : C \to \mathbb{N}$, denotes the estimated time taken for a single visit of the component, computed as:

$$st^a(c_i) = et(c_i) + \sum_{j \in \mathcal{I}} \delta(c_i, c_j) \cdot a(c_i) \tag{1}$$

- Expected number of visits, $v : C \to \mathbb{R}$, quantifies the expected number of visits of a component (subsystem) by system execution, defined as [28]:

$$v(c_i) = q_0(c_i) + \sum_{j \in \mathcal{I}} (v(c_j) \cdot p(c_j, c_i)) \tag{2}$$

Using the above parameters and behavioural description, the system is modelled as a DTMC, where components are represented by the nodes and the interaction among them is represented by the arcs. Redundancy allocation is used as the reliability improvement technique, where adding a new instance of a component has an impact in the cost of the overall system and the communication and processing overhead.

Since the component and interaction parameters in our model can be written as vectors, resp.ṁatrixes, we use matrix operations to solve the above (2). When the transfer probabilities $p(c_i, c_j)$ and execution initiation probabilities of components $q_0(c_i)$ are transformed into matrix notation $P_{n\times n}$ and $Q_{n\times 1}$ respectively. The expected number of visits matrix $V_{n\times 1}$ can be expressed as follows.

$$V = Q + P^T \cdot V$$

By applying usual matrix operations, the above can be transformed into an evaluatable format as,

$$I \cdot V - P^T \cdot V = Q$$

$$(I - P^T)V = Q$$

$$V = (I - P^T)^{-1} \cdot Q$$

One basic requirement of our model is that the architect should provide estimated time for a single visit inside a component. Moreover, redundant components in a subsystem are assumed to be identical and the minimum number of components in a subsystem is 1.

In consideration of redundancy allocation, the model also captures the impact of redundant components to the execution behaviour of the system. This is represented by the estimated additional overheads to the other components that communicate with the duplicated instance. These overheads represent cumulative effect of factors like communication and processing overheads due to synchronisation, consistency check and fault tolerant data transmission to the redundant components.

4 Evaluating Architecture Alternatives

To measure the quality of an architecture alternative a, the presented approach comprises of quality attribute evaluations $Q : A \rightarrow \mathbb{R}^3$, where $Q(a) = (R^a, T^a, C^a)$ for R^a, T^a, C^a defined below.

4.1 Reliability

It is required to compute a metric for reliability in each redundancy allocation alternative a that is generated during the design space exploration. In estimating the reliability of a single component, i.e. an ECU, we assume that failure of a component has an exponential distribution [27], which is characterised by

failure rate parameter λ. Accordingly, the reliability of a single component c_i is evaluated as:

$$R(c_i) = e^{-\lambda \cdot st(c_i)} \tag{3}$$

When the redundancy levels are employed in the model, it is required to quantify the reliability of the composite unit of components, which is referred as a subsystem in section 3. With the assumption of replicating identical components in parallel, the reliability of a subsystem S_i for allocation a can be obtained as:

$$R^a(c_i) = (1 - (1 - R(c_i))^{a(i)+1}) \tag{4}$$

Having calculated the reliability of a subsystem, the next task is to obtain the system reliability metric for a given redundancy allocation function a. As we have modelled the architecture as a DTMC, we employ a well established method of reliability estimation presented by Kubat [28,29]. When the reliability of individual nodes are known in a DTMC model of a system, overall estimation of the system can be computed as follows.

$$R^a \approx \prod_{i \in \mathcal{I}} (R^a(c_i))^{v(c_i)} \tag{5}$$

where $v(c_i)$ represents the expected number of visits of the component during the execution as defined by (2).

4.2 Response Time

In the prediction of response time for each redundancy allocation candidate, we use the DTMC model based approach presented in [14]. Since we describe the system behaviour as a Markov model with probabilities of execution transfer between components together with probabilities of execution initialisation at each component, the expected number of visits can be obtained by (2).

Using (2) and (1) the response time T for a redundancy allocation candidate a can be given as:

$$T^a = \sum_{i \in \mathcal{I}} st^a(c_i) \cdot v(c_i) \tag{6}$$

This metric gives an overall value of response time instead of representing individual values for each trace of execution initiated at different components. Since the expected number of visits for components represents relative usage of them with respect to initialisation probabilities, the above summation (6) indicates a weighted sum of response times for each execution trace from initiator components to the actuators.

4.3 Cost

The cost of the system for each architecture alternative is evaluated as the sum of the costs of individual components and respective redundancies as follows:

$$C^a = \sum_{i \in \mathcal{I}} cst(c_i) \cdot (a(c_i) + 1) \tag{7}$$

5 Architecture Optimisation

To find a trade off between the conflicting quality attributes in $Q : A \rightarrow \mathbb{R}^k$, where k represents the number of attributes, a multi-objective approach should be employed. The redundancy allocation problem referred in this paper is a NP optimisation problem [30]. No NP optimisation problem can be solved in polynomial time, unless P = NP [30]. In these circumstances rather then looking for optimality, which is not feasible, approximate solutions computable in polynomial time are the best choice [30]. Consequently, in this paper we use *Ant Colony Optimisation (ACO)* [31,7] as an optimisation meta-huristic, which is an approximate method already used in the redundancy allocation problem. ACO finds an approximate set of solutions $A^* \subseteq A$ that represent a trade-off among the quality attributes being optimised, i.e. Reliability, Response Time and Cost.

The principle of ACO relies on finding high-quality solutions in a constructive way, by building a probabilistic *pheromone model*. In our redundancy allocation problem, the *pheromone model* is represented by a matrix $PM : C \times N \rightarrow \mathbb{R}$ in which each cell $PM(c_i, rl_j)$ represents the attractiveness of allocating redundancy level rl_j for component c_i. For each quality attribute, a separate pheromone matrix is constructed. The information from the pheromone matrixes is used in such a way that evenly distributed weight combinations are obtained in each iteration. The pheromone matrix is updated with values which are representations of the quality of the constructed solutions. For the selection of new candidates the pseudo-random-proportional rule [31] is employed. The main steps of the algorithm are as follows:

1. The pheromone matrixes are initialised with an initial value $\tau_0 = 1$
2. Every solution is constructed by assigning redundancy levels to all components of the system.
3. The assignment decision is based on the pseudo-random-proportional rule with a 40% greedy approach, as follows:
   3.1 Generate a pseudo random value $q \in [0, 1]$.
   3.2 If $q < 0.4$, select the redundancy level rl^* with the maximal pheromone value $PM(c_i, rl^*) = max_{0<=j<=nMax}$ for the component c_i.
   3.3 Otherwise, select a redundancy level based on the probability defined by the ratio of pheromone values for each level, i.e. redundancy level that has a higher pheromone value has a higher chance to be selected rather than redundancy level that has a lower pheromone value.
4. Local pheromone update is performed after each assignment by reducing the pheromone values of the constructed solutions as follows:

$$\tau \ = \ (1 - \rho) \cdot \tau + \rho \cdot \tau_0 \tag{8}$$

   where ρ is the pheromone evaporation rate equal to 0.1.
5. The quality of the constructed solution is evaluated with Q.
6. The updated population is ranked according to each objective R^a, T^a, C^a.

7. For each objective, the pheromone matrix is updated according to the best and second-best solutions, with $\Delta\tau_b = 10$ for the best and $\Delta\tau_{sb} = 5$ for the second-best solution. The update rule is as follows:

$$\tau \;=\; (1-\rho)\cdot\tau + \rho\cdot\Delta\tau \tag{9}$$

where $\Delta\tau$ is the reward to indicate the quality of the solution.
8. After satisfactory number of iterations, the final approximate set of non-dominated solutions is obtained.

6 Tool Support

The tool support for our approach has been implemented as a part of the *ArcheOpterix* framework [16]. The framework has been developed with Java and Eclipse [32] and can be directly used as a plug-in for the Open Source AADL Tool Environment OSATE [33]. The ArcheOpterix tool provides a generic platform which can be used to specify, evaluate and optimise embedded system architectures. The architecture of the framework is presented in Figure 2, followed by a description of its main elements.

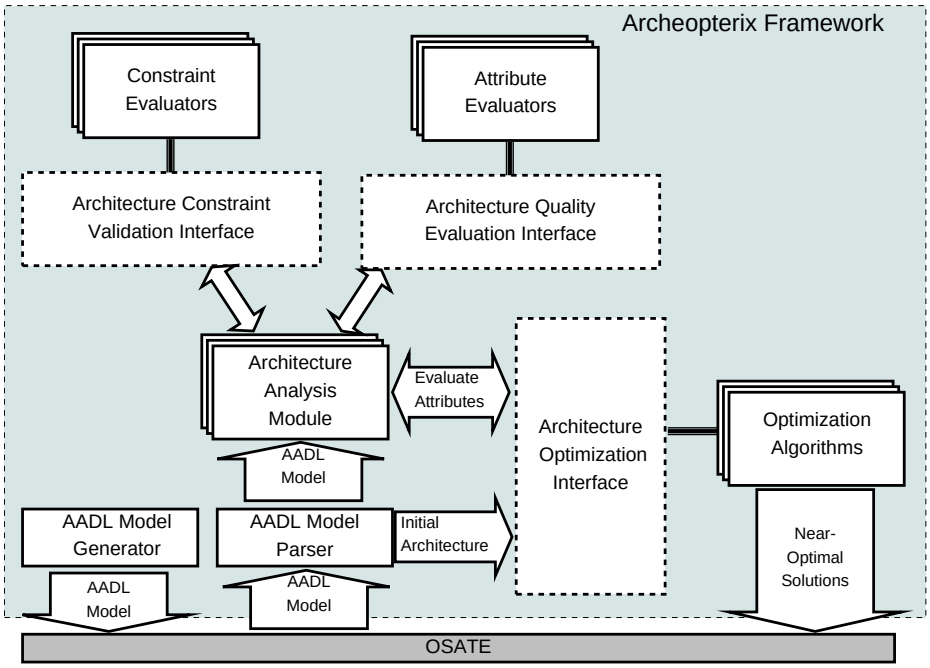

Fig. 2. Architecture of ArcheOpterix framework

ArcheOpterix provides a set of features to specify the embedded-system model (the *AADL Model Parser* module), check for constraint satisfaction (the *Architecture Constraints Validation* interface), evaluate the model (the *Architecture Quality Evaluation* interface) and find the set of near optimal deployments (the *Architecture Optimisation* interface).

AADL Model Parser interprets and extracts the system description from an AADL specification. The module is capable of capturing AADL standard elements such as components, services, processors, busses, etc., and can be extended to other elements and domain-specific parameters. The extracted parameters are used as an input of the *Architecture Analysis Module* which supports the two interfaces for analysing the model, i.e. *Architecture Constraints Validation* and *Architecture Quality Evaluation* interfaces.

Architecture Constraints Validation Interface provides a plug-in point for *Constraint Evaluator* modules that check a given architecture for constraint satisfaction. ArcheOpterix currently implements three *Constraint Evaluator* modules, namely memory, localisation and collocation [34] constraints.

Architecture Quality Evaluation Interface is used to attach different quality evaluation functions. In ArcheOpterix, quality evaluation functions are represented by *Attribute Evaluator* modules, which can be extended with respect to evaluated features. Besides *Reliability* and *Response Time*, the current version of ArcheOpterix implements also *Data Transmission Reliability* and *Communication Overhead*.

Architecture Optimisation Interface provides the means for adding new optimisation algorithms to the framework. The current implementation of the tool comprises *Exact Algorithms*, *Genetic Algorithms* and *Ant Colony Optimisation*.

ArcheOpterix allows translating solutions back into application domain as AADL specifications.

7 Case Study

7.1 Automotive control system

An automotive control system is used as a case study for the demonstration of the approach. The case study has been designed based on already published models [12,35]. We have used a composite system of an automotive *Anti-lock Break System (ABS)* and *Adaptive Cruise Control (ACC)*. Additional parameters required for the model are chosen to closely resemble the reality, like component failure rates [36], estimated execution time per visit [23].

Anti-lock Break System (ABS): The ABS is currently used in most of modern cars to minimise hazards associated with skidding and loss of control due to locked wheels during breaking. Proper rotation during break operations allows better manoeuvrability and enhance the performance of breaking. *Adaptive*

Cruise Control (ACC): Apart from automatic cruise control functionality, the main aim of the ACC is to enable vehicle follow up and avoid crashes by reducing speed once a slower vehicle in front is detected. The main components used by the composite system and their interaction diagram are presented in Figure 3. The *ABS Main Unit* is the major decision making unit regarding the breaking levels for individual wheels, while the *Load Compensator* unit assists with computing adjustment factors from the wheel load sensor inputs. Components 4 and 5 represent the components that communicate with wheel sensors while components 7 and 8 represent the ECUs that control the break actuators. *Break Paddle* is the component that reads from the paddle sensor and sends the data to the *Emergency Stop Detection* unit. Execution initialisation is possible at the components that communicate with the sensors and user inputs. In this case study the *Wheel Sensors*, *Speed Limit*, *Object Recognition*, *Mode Switch* and *Break Paddle* components contribute to the triggering of the service. The captured data from the sensors, will be processed by different components in the system and triggers will be generated for the actuators like *Break Actuators* and *Human Machine Interface*.

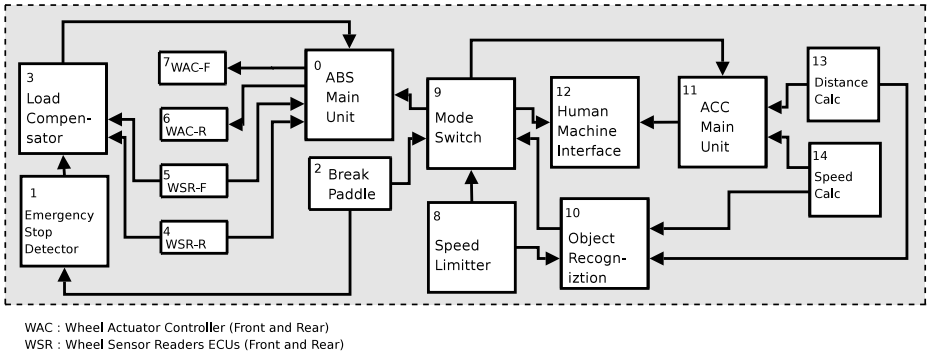

Fig. 3. Automotive composite system

Parameters of the elements of the considered system, and probabilities of transferring execution from one component to another are illustrated in Table 1. The AADL specification of the case study and complete implementation is available on the ArcheOpterix web site[2].

7.2 Results

Even though the presented case study is a comparatively small segment of actual automotive system architecture problem, the possible number of candidate architectures is $4^{15} \approx 1.07 \cdot 10^9$(assuming nMax=3), which is too large to traverse through an exact algorithm. We employed an *Ant Colony Optimisation* algorithm as a guided constructive meta-heuristic to obtain (sub)optimal solutions

[2] `http://www.ict.swin.edu.au/personal/imeedeniya/archeopterix/`

Comp ID	cst $	q_0	λ	et (ms)	δ (ms) (Array for all components)
0	15	0	$4 \cdot 10^{-6}$	50	0,0,0,1,1,1,3,2,0,0,0,0,0,0
1	8	0	$6 \cdot 10^{-6}$	30	0,0,1,3,0,0,0,0,0,0,0,0,0,0
2	10	0.3	$5 \cdot 10^{-6}$	10	0,1,0,0,0,0,0,0,3,0,0,0,0,0
3	10	0	$8 \cdot 10^{-6}$	40	2,2,0,0,2,2,0,0,0,0,0,0,0,0
4	8	0.1	$8 \cdot 10^{-6}$	10	2,0,0,3,0,0,0,0,0,0,0,0,0,0
5	8	0.1	$8 \cdot 10^{-6}$	10	2,0,0,3,0,0,0,0,0,0,0,0,0,0
6	8	0.1	$8 \cdot 10^{-6}$	10	2,0,0,0,0,0,0,0,0,0,0,0,0,0
7	8	0.1	$8 \cdot 10^{-6}$	10	1,0,0,0,0,0,0,0,0,0,0,0,0,0
8	10	0.1	$5 \cdot 10^{-6}$	20	0,0,0,0,0,0,0,0,4,3,0,0,0,0
9	8	0	$5 \cdot 10^{-6}$	20	0,0,2,0,0,0,0,0,0,0,2,2,0,0
10	12	0	$5 \cdot 10^{-6}$	50	0,0,0,0,0,0,0,0,2,1,0,0,0,2,1
11	14	0	$4 \cdot 10^{-6}$	40	0,0,0,0,0,0,0,0,0,0,1,0,3,1,1
12	15	0	$7 \cdot 10^{-6}$	40	0,0,0,0,0,0,0,0,0,1,0,2,0,0,0
13	15	0.1	$3 \cdot 10^{-6}$	50	0,0,0,0,0,0,0,0,0,0,3,1,0,0,0
14	15	0.1	$3 \cdot 10^{-6}$	50	0,0,0,0,0,0,0,0,0,0,2,3,0,0,0

Trans. $c_i \rightarrow c_j$	$p(c_i, c_j)$
$0 \rightarrow 7$	0.5
$0 \rightarrow 6$	0.5
$1 \rightarrow 3$	1
$2 \rightarrow 1$	0.75
$3 \rightarrow 0$	1
$4 \rightarrow 0$	0.7
$4 \rightarrow 3$	0.3
$5 \rightarrow 0$	0.7
$5 \rightarrow 3$	0.3
$2 \rightarrow 9$	0.25
$8 \rightarrow 9$	0.6
$8 \rightarrow 10$	0.4
$9 \rightarrow 0$	0.2
$9 \rightarrow 11$	0.4
$9 \rightarrow 12$	0.6
$10 \rightarrow 9$	1
$11 \rightarrow 12$	1
$13 \rightarrow 10$	0.5
$13 \rightarrow 11$	0.5
$14 \rightarrow 10$	0.5
$14 \rightarrow 11$	0.5

Table 1. Parameters of Components and Interactions

in practically affordable time frame. The parameters of the algorithm have been used as presented in the section 5.

The execution of the algorithm was set to 50 000 candidate evaluations, and performed under a settings on a dual-core 2.26 GHz processor computer. The algorithm took 82 seconds for the 50 000 function evaluations and generated 48 non-dominated solution architectures. The distribution of the (near)optimal solutions which are graphically represented in Figure 4. The prevalence of the solutions in the objective domain, together with their (sub)optimal trade-offs are depicted in the graph, the designer will be privileged to make decisions on suitable candidate solution.

Solution	Redundancy Allocation	Reliability	Cost ($)	Res. Time(ms)
A	$[1, 3, 3, 2, 0, 0, 1, 1, 2, 2, 1, 1, 1, 1, 0]$	0.999968946	361	133.1950073
B	$[1, 3, 0, 2, 0, 0, 1, 2, 1, 1, 2, 2, 1, 1, 1]$	0.999968946	362	129.9750061

Table 2. Example non-dominated solutions

Table 2 illustrates two closely related non-dominated solutions generated by the optimisation process. The arrays in the second column represents the redundancy levels for the components in the order of their ID. Note that the reliability of both the solutions are identical and they are competitively better in

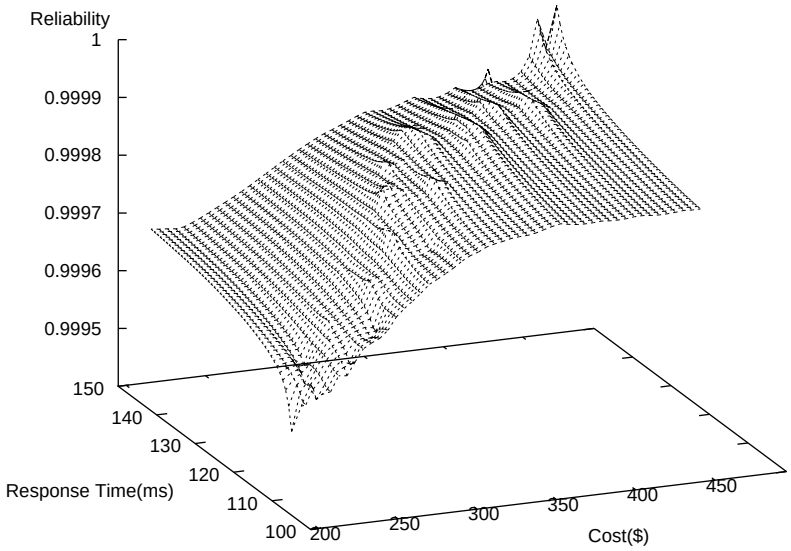

Fig. 4. Distribution of (near)Pareto solutions

ID	Component	Solution A	Solution B	Exp.Visits
2	Load Compensator	3	0	0.3
7	Actuator Control (Rear)	1	2	0.3
8	Speed Limiter	2	1	0.1
9	Mode Switch	2	1	0.3
10	Object Recognition	1	2	0.1
11	ACC Unit	1	2	0.2
14	Speed Detector	0	1	0.1

Table 3. Comparison of the two solutions

other two objectives. The differences in the redundancy levels in the two solutions are presented in table 3. The column title "Exp.visits" follows the definition of expected number of visits presented in formula 2. It should be highlighted that, the non-obvious changes from solution A to solution B has significantly reduced the response time for a trade-off on small cost while keeping the system reliability at the same level.

8 Conclusions and Future Work

Conclusions In this paper, we address the redundancy allocation problem in embedded system architectures, focused on the nature of the problem in automotive industry. The approach was capable of assisting the system architecture designer providing (near)optimal set of solutions with respect to reliability, response time and cost. We have employed an *Ant Colony Optimisation* algorithm to guide the search and implementation has been carried out by extending the

ArcheOpterix framework. Reliability, Response Time and Cost evaluation for the model has been done by appropriate use of approaches from the literature. An automotive case study of a composite system of *Anti-lock Break System* and *Adaptive Cruise Control* has been used for the validation. A near-optimal set of possible redundancy allocation decision has been obtained.

Future Work One key aspect in this paper is that the components are modelled as COTS that are composites of both software and hardware. We expect to extend the approach in the case of general purpose ECUs and deploying software components on them. Solving the described redundancy allocation problem together with obtaining optimum resource usage via deployment would be one of our key future focus. In obtaining more realistic view on reliability for automotive embedded systems, capturing the reliability implications to inter-component communication is also important. We also wish to extend the approach by using non-homogeneous components as redundancies. When the architect is provided with a set of component selection alternatives, obtaining the best possible design trade-offs with respect to user preferences will be also considered.

Acknowledgement

This original research was proudly supported by the Commonwealth of Australia, through the Cooperative Research Centre for Advanced Automotive Technology (projects C4-501: Safe and Reliable Integration and Deployment Architectures for Automotive Software Systems and C4-509: Dependability optimisation on architectural level of system design), and the Academy of Sciences of the Czech Republic (grant No. 1ET400300504).

References

1. K. Grimm, "Software technology in an automotive company: major challenges," in *Proceedings of the 25th International Conference on Software Engineering (ICSE-03)*, pp. 498–505, IEEE Computer Society, 2003.
2. A. Möller, M. Åkerholm, J. Fredriksson, and M. Nolin, "Evaluation of component technologies with respect to industrial requirements," in *30th EUROMICRO Conference*, pp. 56–63, IEEE Computer Society, 2004.
3. M. Åkerholm, J. Fredriksson, K. Sandström, and I. Crnkovic, "Quality attribute support in a component technology for vehicular software," in *Fourth Conference on Software Engineering Research and Practice*, 2004.
4. J. Fredriksson, M. Tivoli, and I. Crnkovic, "A component-based development framework for supporting functional and non-functional analysis in control system design," in *ASE:20th IEEE/ACM International Conference on Automated Software Engineering (ASE 2005)*, pp. 368–371, 2005.
5. "Autosar development partnership." `http://www.autosar.org`.
6. L. Grunske, "Early quality prediction of component-based systems - a generic framework," *Journal of Systems and Software*, vol. 80, no. 5, pp. 678–686, 2007.

7. Y.-C. Liang and A. E. Smith, "An ant system approach to redundancy allocation," in *1999 Congress on Evolutionary Computation*, pp. 1478–1484, IEEE Service Center, 1999.
8. D. W. Coit and A. E. Smith, "Solving the redundancy allocation problem using a combined neural network/genetic algorithm approach," *Computers & OR*, vol. 23, no. 6, pp. 515–526, 1996.
9. S. Kulturel-Konak, D. W. Coit, and F. Baheranwala, "Pruned pareto-optimal sets for the system redundancy allocation problem based on multiple prioritized objectives," *J. Heuristics*, vol. 14, no. 4, pp. 335–357, 2008.
10. L. Grunske, P. A. Lindsay, E. Bondarev, Y. Papadopoulos, and D. Parker, "An outline of an architecture-based method for optimizing dependability attributes of software-intensive systems," in *Workshop on Architecting Dependable Systems (WADS)'06*, vol. 4615 of *Lecture Notes in Computer Science*, pp. 188–209, Springer, 2006.
11. V. S. Sharma, P. Jalote, and K. S. Trivedi, "Evaluating performance attributes of layered software architecture," in *Proceedings of the 8th International Symposium of Component-Based Software Engineering, CBSE 2005*, vol. 3489 of *Lecture Notes in Computer Science*, pp. 66–81, Springer, 2005.
12. J. Fredriksson, T. Nolte, M. Nolin, and H. Schmidt, "Contract-based reusableworst-case execution time estimate," in *3th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007)*, pp. 39–46, 2007.
13. S. Islam and N. Suri, "A multi variable optimization approach for the design of integrated dependable real-time embedded systems," in *Embedded and Ubiquitous Computing, International Conference, EUC 2007, Proceedings*, vol. 4808 of *Lecture Notes in Computer Science*, pp. 517–530, Springer, 2007.
14. V. S. Sharma and K. S. Trivedi, "Quantifying software performance, reliability and security: An architecture-based approach," *Journal of Systems and Software*, vol. 80, no. 4, pp. 493–509, 2007.
15. C. A. C. Coello, "Evolutionary multi-objective optimization: a historical view of the field," in *Computational Intelligence Magazine, IEEE*, pp. 28–36, 2006.
16. A. Aleti, S. Björnander, L. Grunske, and I. Meedeniya, "ArcheOpterix: An extendable tool for architecture optimization of AADL models," in *MOMPES*, pp. 61–71, ACM and IEEE Digital Libraries, 2009.
17. D. W. Coit and A. E. Smith, "Reliability optimization of series-parallel systems using a genetic algorithm," *IEEE Transactions on Reliability*, vol. 45, pp. 254–260, 1996.
18. S. Kulturel-Konak, A. E. Smith, and D. W. Coit, "Efficiently solving the redundancy allocation problem using tabu search," *IIE Transactions*, vol. 35, no. 6, pp. 515–526, 2003.
19. L. Grunske, "Identifying "good" architectural design alternatives with multi-objective optimization strategies," in *28th International Conference on Software Engineering (ICSE 2006)*, pp. 849–852, ACM, 2006.
20. S. Islam, N. Suri, A. Balogh, G. Csrtan, and A. Pataricza, "A transformation based design for integrated dependable real-time embedded systems," in *TODO:In Review*, pp. 0–0, 2009.
21. S. Becker, H. Koziolek, and R. Reussner, "Model-based performance prediction with the palladio component model," in *Proceedings of the 6th International Workshop on Software and Performance, WOSP 2007*, pp. 54–65, ACM, 2007.

22. Y. Papadopoulos and C. Grante, "Techniques and tools for automated safety analysis & decision support for redundancy allocation in automotive systems," in *27th International Computer Software and Applications Conference: Design and Assessment of Trustworthy Software-Based Systems (COMPSAC 2003)*, IEEE Computer Society, 2003.
23. B. Florentz and M. Huhn, "Embedded systems architecture: Evaluation and analysis," in *Quality of Software Architectures, Second International Conference on Quality of Software Architectures (QoSA 2006)*, vol. 4214 of *Lecture Notes in Computer Science*, pp. 145–162, 2006.
24. D.-J. Chen, R. Johansson, H. Lönn, Y. Papadopoulos, A. Sandberg, F. Törner, and M. Törngren, "Modelling support for design of safety-critical automotive embedded systems," in *27th Internationa Conference of Computer Safety, Reliability, and Security (SAFECOMP 2008), Proceedings*, vol. 5219 of *Lecture Notes in Computer Science*, pp. 72–85, Springer, 2008.
25. "Advancing traffic efficiency and safety through software technonlogy." `http://www.atesst.org`.
26. K. S. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. Prentice-Hall, 1982.
27. S. M. Shatz, J.-P. Wang, and M. Goto, "Task allocation for maximizing reliability of distributed computer systems," *IEEE Transactions on Computers*, vol. 41, no. 9, pp. 1156–1168, 1992.
28. P. Kubat, "Assessing reliability of modular software," *Operations Research Letters*, vol. 8, no. 1, pp. 35–41, 1989.
29. K. Goseva-Popstojanova and K. S. Trivedi, "Architecture-based approach to reliability assessment of software systems," *Performance Evaluation*, vol. 45, no. 2-3, pp. 179–204, 2001.
30. P. Crescenzi and V. Kann, "A compendium of NP optimization problems," 2000.
31. M. J. Csorba, P. E. Heegaard, and P. Herrmann, "Cost-efficient deployment of collaborating components," in *8th International Conference in Distributed Applications and Interoperable Systems (DAIS 2008), Proceedings*, vol. 5053 of *Lecture Notes in Computer Science*, pp. 253–268, 2008.
32. F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, and T. Grose, *Eclipse Modeling Framework*. Addison Wesley Professional, 2003.
33. "Open source aadl tool environment (osate)." `http://la.sei.cmu.edu/aadlinfosite/AADLTools.html`.
34. N. Medvidovic and S. Malek, "Software deployment architecture and quality-of-service in pervasive environments," in *Proceedings of the 2007 International Workshop on Engineering of Software Services for Pervasive Environments, (ESSPE 2007)*, pp. 47–51, ACM, 2007.
35. L. Grunske, "Towards an Integration of Standard Component-Based Safety Evaluation Techniques with SaveCCM," in *Quality of Software Architectures, QoSA 2006,*, vol. 4214 of *LNCS*, pp. 199–213, Springer, 2006.
36. I. Assayad, A. Girault, and H. Kalla, "A bi-criteria scheduling heuristic for distributed embedded systems under reliability and real-time constraints," in *Dependable Systems and Networks (DSN 2004)*, pp. 347–356, IEEE Computer Society, 2004.