ACHIEVING TRADEOFFS BETWEEN FUNCTIONALITY AND SAFETY IN EARLY SYSTEM DESIGN

Christian Grante[1], Yiannis Papadopoulos[2]

[1]*Volvo Cars, Sweden, cgrante@volvocars.com*
[2]*Department of Computer Science, University of Hull, U.K., y.i.papadopoulos@hull.ac.uk*

Abstract: In this paper, we propose a novel method that helps designers to form design concepts that achieve optimal tradeoffs among functionality, cost and safety early in the design. The method combines genetic algorithms with preliminary risk analysis and is largely automated thus satisfying the preconditions for application in complex systems. A Pareto multi-objective optimisation approach helps to generate a set of design concepts that represent optimal solutions for different levels of expenditure and risk, while the final decision on which design concepts are most promising is left to humans.

Keywords: risk, reliability analysis, safety critical systems, multi-objective optimization, genetic algorithms

.

1. INTRODUCTION AND BACKGROUND

The design of a new engineering system (e.g. a car) typically starts at a conceptual level where the functions to be provided by the system must be decided in the context of constraints like the cost of components, the cost of development and the production capabilities of a company. There are usually a number of design possibilities that can be translated to different concepts which could provide different functions. Issues like potential markets, volumes, costs, reliability or safety requirements and ultimately the potential for profit have to be addressed for a concept to be translated into a successful design. Typically, satisfaction of most constraints is required; however the motivation for proceeding further with a design concept is usually the potential for profit measured absolutely or more commonly as return on investment.

There is often a wealth of information from past experience or competitor products upon which the decision about the potential of a new design could be evaluated. Such information typically includes knowledge of desirable functions, of the value that customers are prepared to pay for those functions, the components that are likely to be needed to deliver such functions and in many case reliability data about components. In practice, however, it is not always obvious how this diverse information can be used to decide which combination of functions and components should be selected for the new design. This is especially true when there are a vast number of viable combinations of functions and components, because a decision on which combination is best would require evaluation of all those design options using, for example, simple or more complex calculations of profit, cost and perhaps reliability or risk. Realistically, however, and for pragmatic designs which may have hundreds of functions and thousands of components, exhaustive evaluation of all options is impossible even if the simplest calculations were adopted. This is especially true in designs for complex distributed systems where functions can be allocated in many different ways on components of the architecture resulting to a plethora of different design concepts.

To address this problem, in this paper we present a method that provides automated support to designers in the difficult task of arriving at an optimal *design*

concept for a complex system at early stages of the design. The term *design concept* is used here to describe a set of functions delivered by the system and a set of components that represent the *technical implementations*[1] of these functions. An *optimal* design concept is one that provides maximum potential for profit and can be achieved within budget whilst leading to an implementation that offers acceptable levels of safety. Such design concepts would clearly be desirable representing potentially feasible and socially acceptable business opportunities in which it would be worth investing.

2. OUTLINE OF METHOD

In the proposed approach, the design of a new engineering system starts with the construction of a list of functions considered for inclusion in the new system. For a system that provides active safety functions in a passenger car, for instance, such functions would include antilock braking, traction control, emergency brake assistance, and vehicle stability control. In the proposed method, the following information should be established for each function potentially included in the system:

- an estimate of the extra value that customers would be prepared to pay for inclusion of this function
- the technical implementation of the function, i.e. a set of components needed to deliver the function

Note that in our approach, each component can participate in more than one technical implementation. The relationship between functions and components must therefore be established and provided as an input in the form of a matrix that relates functions to sets of components (i.e. their technical implementations). For each component potentially employed by a function, the cost and average failure frequency must also be established and provided as input to the optimisation process. We hope that this requirement would not pose a problem in practice, as large companies are reasonably expected to maintain cost and reliability databases that are already useful in existing applications.

The set of *all* candidate functions considered for a new design obviously engages all candidate components and therefore represents the most expensive (and functionally powerful) potential design solution. Each subset of these functions represents a less complete and less expensive specification for the system under design. Given that data about the value of functions and the cost & reliability of components is available, then the potential profit, cost and level of safety associated with each potential design solution can be calculated as follows:

- Cost can be calculated as the sum of costs of components participating in all technical implementations of the included functions.
- Profit can be calculated as the sum of function values minus the overall cost of components.
- An indication of the safety offered by each design solution can be given by a preliminary calculation of risk[2]. Risk can, in turn, be estimated as the sum, calculated over all functions in a design solution, of the severity of functional failure multiplied by the failure frequency of the respective technical implementation.

Assuming that data is available to perform the above calculations, the problem that we try to address can then be seen as one of identifying which design solution provides optimal values for the above calculations of cost, profit and risk. One approach to this problem would be to exhaustively enumerate all potential design solutions (representing different combinations of functions) and for each solution perform the above calculations. However, if we were to adopt this approach, we would encounter a classical problem of combinatorial explosion, which in practice means that for a modestly large number of functions the number of design solutions and consequent calculations is too large to contemplate.

To overcome this difficulty, we opted to apply evolutionary search and optimisation techniques. We developed a genetic algorithm which performs a systematic, but selective search for those potential design solutions (i.e. configurations of functions and components) that optimise profit within given cost constraints and risk requirements. The algorithm simultaneously generates several solutions by carrying out an iterative multidirectional search. In the course of this search, the genetic algorithm calculates the fitness of candidate designs in terms of profit, cost and risk and then ranks potential solutions according to their "degree of dominance" (Fonseca and Fleming, 1998) thus forming and progressively improving a "Pareto front" of optimal solutions (see section 4 for details).

In the context of the problem that we examine, this type of multidirectional search provides the essential flexibility required in the final decision making. Theoretically optimal (in terms of profit, cost and risk) solutions, for example, may be impractical because they imply awkward physical arrangements of components. In such cases, suboptimal solutions may be preferable. In a different scenario, solutions may be theoretically better than others because they achieve higher profit within the same cost constraint. However, the extra profit may be achieved at significant extra cost in which case less profitable designs may be preferable as they achieve a better ratio of profit over cost. Precisely because such decisions are very difficult, rather than prescribing a

[1] For the purposes of this paper, the term *technical implementation* is used to describe the set of components needed for the delivery of the function

[2] Safety can indeed be defined as "freedom from unacceptable risk", see for example (CENELEC, 1999)

single optimal solution, the proposed method generates several solutions and leaves the final judgements to humans.

A precise mathematical formulation of the problem and a more detailed presentation of the optimisation algorithms follow which, we hope, provide a more thorough exposition of the proposed approach.

3. MATHEMATICAL FORMULATION

To represent the problem more formally, a matrix notation was developed drawing from earlier work by Suh (2001) in axiomatic design and Grante and Andersson (2003) in design optimisation. In this notation, functions delivered by the system under design are represented in a function vector **F** and components in a component vector **C**. A realisation vector $\mathbf{rv_i}$ defines the technical implementation of function F_i. Each element of the realisation vector declares the presence or not of a component in the technical implementation and can have the value of one – indicating that the corresponding component is needed – or zero indicating that the component is not needed. The realisation vectors for all customer functions,

$$\mathbf{rv}_i = 1 \ldots m,$$

together form the realisation matrix **RM**. Thus the problem can be described according to equation (i),

$$\mathbf{F} = \mathbf{RM} \cdot \mathbf{C} \qquad (\text{i})$$

i.e.,

$$\begin{bmatrix} F_1 \\ \cdots \\ \cdots \\ F_m \end{bmatrix} = \begin{bmatrix} - & \mathbf{rv}_1 & - \\ & \cdots & \\ & \cdots & \\ - & \mathbf{rv}_m & - \end{bmatrix} \begin{bmatrix} C_1 \\ \cdots \\ \cdots \\ C_n \end{bmatrix}$$

where m is the number of potential functions and *n* the number of components. The function vector differs from the component vector in that the component vector may have alternative implementations of components, for example variants provided by different manufacturers.

Each combination of components yields one possible solution to the problem. For a problem with *n* components and an average of *d* alternative implementations for each component there are $2^{(n*d)}$ different possible solutions (note the combinatorial explosion). A particular solution, **X**, is expressed by a vector

$$\mathbf{X} = \left[x_1, x_2, \ldots . x_n \right],$$

where x_i is either zero indicating that the component is not part of the design solution or a natural number between 1 and d_{TIi} indicating which implementation of component is part of the solution. In order to calculate the value that customers will be prepared to pay for a specific solution, the functions that could be possibly realised using the components included in a solution must be determined. The function realisation vector, **W**, represents this. W is calculated according to equation (ii).

$$\mathbf{W}(\mathbf{X}) = \begin{bmatrix} w_1 \\ \ldots \\ \ldots \\ w_m \end{bmatrix}$$

where

$$w_i(\mathbf{X}) = \left\lfloor \frac{\mathbf{rv}_i \cdot (diag\mathbf{X}^T \cdot diagin\mathbf{X}^T)}{\mathbf{1}^T \cdot \mathbf{rv}_i^T} \right\rfloor \qquad (\text{ii})$$

Where $diag\mathbf{X}^T$ is a diagonal matrix with the elements of $\mathbf{X}^T$ in the diagonal and $diagin\mathbf{X}^T$ is a diagonal matrix with the elements of $\mathbf{X}^T$ which are not zero inverted in the diagonal. Thus,

$$diag\mathbf{X}^T \cdot diagin\mathbf{X}^T$$

is a vector that contains ones and zeros, where one indicates that a component is a part of the solution and a zero that it is not. The notation

$$\lfloor a \rfloor$$

denotes the largest integer that is less than or equal to *a*. The number of components used by function *i* that are included in solution **X** is represented by

$$\mathbf{rv}_i \cdot (diag\mathbf{X}^T \cdot diagin\mathbf{X}^T).$$

This number is divided by the number of all components needed to implement function *i*, i.e.

$$\mathbf{1}^T \cdot \mathbf{rv}_i^T,$$

where $\mathbf{1}^T$ is a vector of one. If **X** contains all components needed by F_i this quotient equals 1, otherwise it is less than one. Equation (ii) thus returns 1 only for functions that are implemented by **X**. The total value, *tv*, of a solution is calculated by adding the values of all functions that can be realised by solution **X**, i.e.

$$tv(\mathbf{X}) = \mathbf{W}(\mathbf{X})^T \cdot \mathbf{V} \qquad (\text{iii})$$

where **V** is the vector containing the values that customers are prepared to pay for functions 1..m respectively, i.e.,

$$\mathbf{V}^T = \left[v_1, v_2, \ldots, v_m \right].$$

The cost of implementing each component is represented by the cost-implementation matrix **CIM**,

$$\mathbf{CIM} = \begin{bmatrix} c_{11} & c_{21} & \cdots & c_{1n} \\ c_{12} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{1a1} & c_{2a2} & \cdots & c_{nan} \end{bmatrix}$$

where each component cost (c_{ij}) contains the development, material and production costs. Note that each component n may have up to a_n different implementations. **CIM** is therefore a representation of the costs involved in these alternative implementations. The total cost c of solution **X** is obtained by adding the cost of the selected implementations for components participating in **X** (components for which $x_i = 1$). In this simplified model, profit, p, is expressed as the value that the customer is prepared to pay for a particular design solution minus the cost of developing and producing it; see equation (iv).

$$p(\mathbf{X}) = tv(\mathbf{X}) - c(\mathbf{X}) \qquad (\text{iv})$$

The optimisation problem can thus be described as finding the set of components, **X**, that maximises profit p(**X**) without exceeding the development budget. To introduce safety as a parameter in the optimisation, we also employ the commonly used definition of safety as "freedom from unacceptable risk" and introduce risk calculations into the model. Risk is defined as a combination of the consequence (*s)* and the frequency (*f*) of an unplanned, undesirable event. According to Thomson (Thomson, 1987) but also the CENELEC railway safety standards (CENELEC, 1999), risk is the product of consequence and frequency, i.e.

$$risk = f \times s \qquad (\text{v})$$

Since several events can occur, the risk is the sum of all events, and for n events the risk is calculated according to equation (vi).

$$risk = \sum_i^n (f)_i \times (k)_i \qquad (\text{vi})$$

A product introduced on the market has to be safe, i.e. must only entail an acceptable level of risk. Thus, in order to evaluate the risk of a potential design solution, additional data must be included in the model. The first such data is the consequence vector

$$\mathbf{H} = [h_1, h_2, \ldots, h_m],$$

which contains a quantified indication of the severity of failure of each function. For guidelines on how to interpret qualitative severity classes (marginal, critical, catastrophic) into quantitative indices the reader is referred to the CENELEC standards (CENELEC,1999)

The second additional input required to enable risk calculations is the matrix,

$$\mathbf{FM} = \begin{bmatrix} f_{11} & f_{21} & \cdots & f_{1n} \\ f_{12} & f_{22} & \cdots & f_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ f_{1a1} & f_{2a2} & \cdots & f_{nan} \end{bmatrix}$$

that contains the frequency of malfunction of component implementations, which is similar in structure to the cost-implementation matrix **CIM**.

The main difficulty in performing preliminary risk assessment so early in the design is that it must be done in the absence of a system architecture. This is inevitable, though, since the aim is precisely to establish which functions the design should include, and from this to proceed in the development of an architecture. Without knowledge about how components are connected, however, risk cannot be calculated. For the purposes of this analysis, a conservative *series* architecture is always assumed in which functions fail if any single component in their technical implementation fails. We should note that although this is a conservative assumption, it often gives a fair estimation of the actual behaviour of functions when they are eventually realised in systems. If fault tolerance is already planned at this early stage, fault tolerant schemes will have to be represented as single components with the reduced failure frequency that they achieve directly entered in the frequency matrix **FM**. From the data specified above, and using classical reliability theory, the failure frequency of each function ff_i, can be calculated according to equation (vii).

$$ff_i = 1 - \prod_{j=1}^{n} (1 - f_j), \forall j \qquad (\text{vii})$$

where f_j is the failure frequency of the selected implementation of each component. If the component does not participate in the design solution ($x_i = 0$) then f_i is zero. A function frequency vector can now be created to include the failure frequencies of all functions according to (viii).

$$\mathbf{FF} = [ff_1, ff_2, \ldots, ff_m] \qquad (\text{viii})$$

The level of risk associated with each function, cfr_i, is obtained by multiplying the frequency of malfunction with the severity of consequence:

$$fr_i = h_i \cdot ff_i \qquad (\text{ix})$$

A new vector of function risk **FR** can now be obtained according to (x), where diag(H) represents a diagonal matrix with the elements of H in the diagonal.

$$\mathbf{FR} = diag(\mathbf{H}) \cdot \mathbf{FF} \qquad (\text{x})$$

Finally, the total risk, r, of a design concept is calculated as the sum of all risks associated with the functions included in this concept. Thus r is calculated by multiplying the customer function risk vector with the function realisation vector **W**(**X**), see equation (xi).

$$r(\mathbf{X}) = (X)\mathbf{FR}^T \cdot \mathbf{W}(\mathbf{X}) \qquad (\text{xi})$$

This equation can be used for prediction of the likely risk associated with a potential design solution.

4. EVALUATION USING OPTIMISATION

Assuming that input data is provided for functions and components according to the specification of section 3, the problem is then formulated as one of multiple-objective optimisation. The objectives are to maximise the likely profit of a design concept and simultaneously minimise cost and risk as specified below:

$$\begin{aligned} &\max_X \ p(\mathbf{X}) \\ &\min_X \ c(\mathbf{X}) \\ &\min_X \ r(\mathbf{X}) \qquad \text{(xii)} \\ &\text{s.t.} \quad \mathbf{X} = [x_1, x_2, \ldots, x_n] \\ &\qquad x_i = 0 \vee 1, \forall i \{1,2,\ldots,n\} \end{aligned}$$

Beyond the above formulation of the problem as multi-objective optimisation, we also opted for a multi-directed search in which the optimisation algorithm is looking not for a single optimal solution but for a set of Pareto optimal solutions also known as *non-dominated* solutions. In Pareto optimisation a solution is said to dominate another if it is better in all objectives. Thus, if we consider a minimisation problem with *k* objectives and two solution vectors, **x** and **y**, then **x** is said to dominate **y**, denoted $\mathbf{x} \succ \mathbf{y}$, if:

$$\begin{aligned} &\forall i \in \{1,2,...,k\}: f_i(\mathbf{x}) \leq f_i(\mathbf{y}) \text{ and} \\ &\exists j \in \{1,2,...,k\}: f_j(\mathbf{x}) < f_j(\mathbf{y}) \qquad \text{(xiii)} \end{aligned}$$

Equation (xiii) implies that **x** dominates **y** if x is as good as **y** in all objectives, and that there is at least one objective in which **x** is better. Using this concept of non-dominated solutions, our approach yields a set of design solutions that provide higher potential for profit for different levels of expenditure and risk. The result of this optimisation can be graphically visualised as a Pareto front which shows trade-offs between profit, cost and risk, i.e. how maximum profit increases as cost and safety constraints are decreased in various configurations and vice versa.

The combination of a Pareto approach with genetic algorithms has previously been shown to work well on similar engineering design problems, see (Andersson, 2001), and was, therefore, also adopted in this work. To address the given optimisation problem, first it was necessary to model potential design solutions as genetic material (chromosomes) that can be meaningfully manipulated by a genetic algorithm. For this purpose, a string of natural numbers with the same length as the number of available components was used to represent each potential design solution. The value of each number in the string is 0 when the corresponding component does not participate in the design solution or a positive integer that corresponds to the variant of the implementation of the component employed in the solution.

We also developed a genetic algorithm which applies an evolutionary optimisation process on this type of genetic material. In the course of that process, the algorithm first creates a number of random chromosomes each representing a potential design solution that employs some of the candidate functions and components. The algorithm then calculates the fitness of each individual in this population of individual solutions using the mathematical model presented in section 3. One difficulty here is that, in Pareto optimisation, there is no single objective function to determine the fitness of different individuals. Therefore, the relative ranking scheme presented by Fonseca and Fleming (Fonseca and Fleming, 1998) is used to rank individuals according to their "degree of dominance" which for each individual equals the number of individuals that is dominated by, plus one. In this approach, design concepts with a degree of dominance equal to one are effectively non-dominated solutions which lie on the Pareto front, and therefore represent the fittest designs in a given population of candidate design solutions.

To progressive improve this Pareto front in the course of the evolutionary optimisation, the genetic algorithm creates new generations of candidate design solutions using an implementation of a recently proposed algorithm (Anderson and Wallace, 2002). In this algorithm, parents are chosen from the two most recent generations of candidate designs and then basic genetic mixing and modification mechanisms such as one-point crossover and flip mutation are performed to create children. For each child, the most genetically similar individual in the entire population is then identified and the fitness of this individual is compared with that of the child. If the child is fitter it replaces the older individual in the population.

There is evidence that this replacement strategy counteracts genetic drift that can confine population diversity and lead to inbreeding. Poor diversity in practice means that the population is clustered at the extremes of the Pareto front, which means that trade-offs that could be achieved in the middle area are not clearly identified. Note that in classical genetic algorithm optimisation, the population of a genetic algorithm usually converges to a single optimal point [Goldberg, 1989]. However, using the concept of dominance it is possible to adjust the algorithm so that it spreads the population effectively over the entire Pareto front, thus helping to identify tradeoffs among the different parameters of the optimisation.

5. RESULTS

The method has been tested in Volvo Cars on a case study which included 52 active safety functions supported by 48 components for a new vehicle model. This configuration could result in more than 70 billion possible design concepts. Manual evaluation of all those concepts with regards to cost, profit and risk would have certainly been impossible. However, with the aid of the optimisation tool, it was

possible to arrive at a smaller set of concepts that potentially maximised profit within given cost and risk constraints. A number of interesting conclusions can be drawn from this study.

Firstly, the study showed that the functionality (i.e. the value for customers) and potential profit of the system almost doubled when the optimisation approach was used as a replacement of the traditional manual approach to development and selection of new design concepts.

Secondly, when only cost and profit criteria were used in the optimisation process many optimal concepts entailed an unacceptable level of risk (see the concepts in Fig.1 marked as circles). If these design concepts were allowed to be further developed, their design would almost certainly have been revisited at a later stage to address the problem of high risk.

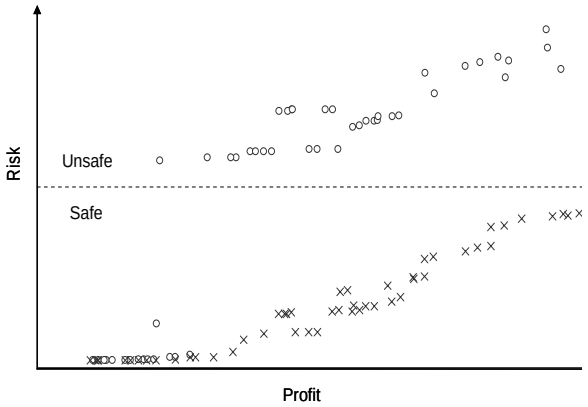

Fig. 1. Risk of design concepts before (o) and after (x) the design iteration aided by the tool.

However, by introducing risk as a parameter in the optimisation process and by iterating the method we were able to identify and correct design weaknesses early in the design. Such weaknesses included key components that consistently contributed to a substantial increase of risk across the design space. By replacing these components with more reliable equivalents a significant risk reduction was achieved in almost all concepts (crosses in Fig.1). The consequent shift of the Pareto front in the area of acceptable risk is illustrated in Fig.1.

6. CONCLUSIONS

There is presently a lack of methods to support the development and optimisation of abstract design concepts for new systems. As systems become more complex, especially distributed systems, the absence of such support becomes increasingly problematic.

In this paper, we presented a method that provides automated support in the optimisation of abstract designs with respect to a number of objectives that include cost, profit and risk. A mathematical framework was presented which models the relationships between functions and their technical implementations. Based on this framework, the problem of developing abstract design concepts was formulated as a multi-objective optimisation problem, i.e. one of maximising profit while keeping to a restricted development budget and creating a product that entails only acceptable levels of risk.

An optimisation tool was developed, and, with the aid of this tool, the method was evaluated in a case study on active safety systems, performed by Volvo Car Corporation. A main result from this study is a Pareto front of optimal design solutions for an active safety system that explores potential trade-offs between profit, cost and risk in various configurations for this system. The study has shown that the proposed method could double the customer value and company returns for given levels of expenditure. It also demonstrated the potential of the method for initiating useful design iterations driven by risk criteria at very early stages of the design.

By using this approach, we hope that some aspects of early design can be improved. At the same time, we currently extend this work by developing a concept for application to more refined architectural models produced at later stages of the design (Papadopoulos and Grante, 2004). The potential benefits from further extension of these techniques are substantial and include further rationalisation of the design process, fewer late design changes, improved value for customers and potentially higher returns for producers.

REFERENCES

Andersson, J. (2001). *Multiobjective Optimization in Engineering Design*, Dissertation, Department of Mechanical Engineering, Linköping University, Linköping.

Andersson, J. and D. Wallace (2002). Pareto Optimization Using the Struggle Genetic Crowding Algorithm, *Engineering Optimization* **34** (6):623-643.

CENELEC (1999). *Railway applications: The Specification and Demonstration of Dependability, Reliability, Availability, Maintainability and Safety*, EN-50126, European Committee for Electro-technical Standardisation.

Grante, C. and J. Andersson (2003). Optimisation of Design Specifications for Mechatronic Systems, *Research in Engineering Design*, **14** (4):224-235.

Goldberg, D.E. (1989). *Genetic Algorithm in Search and Machine Learning*, Addison Wesley.

Fonseca, C. and P. Fleming (1998). Multi-objective Optimization and Multiple Constraint Handling with Evolutionary Algorithms, *IEEE Trans. on Systems, Man, & Cybernetics*, **28**:26-37.

Papadopoulos, Y. and C. Grante (2004). An Evolutionary Process for the Design of Safe Computer-based Systems, INCOM 2004, Salvador, Brasil.

Suh, N. (2001). *Axiomatic Design Advances and Applications*, Oxford University Press, New York.

Thomson, J.R. (1987). *Engineering Safety Assessments*, Longman Scientific & Technical, Longman Group UK Limited.