Optimizing System Requirements with Genetic Algorithms

Alistair Sutcliffe, Wei-Chun Chang, Richard Neville
Centre for HCI Design,
Department of Computation
UMIST, PO Box 88, Manchester, M60 1QD
Email: mcaijwc2@co.umist.ac.uk

Abstract - We introduce evolutionary computation into requirements engineering and propose process for selecting an optimal set of components to satisfy fitness criteria of system reliability. The system architecture and algorithms are described with a case study of a command and control system. The results of the evolutionary requirements analysis are discussed.

Keywords: requirements engineering, scenarios, system requirements, genetic algorithms, evolutionary computation

I. INTRODUCTION

In systems development there is a pressing need to select an optimal set of system requirements at an early stage of the systems engineering process; however selecting optimal requirements is an error prone and labour intensive process. One solution to the problem of defining an optimal set of system requirements is to automatically search the problem domain in order to select the most reliable solution. Artificial intelligence utilizes genetic algorithms (GAs) as stochastic search algorithms to search a problem's state space and find an optimal solution to real world problems by optimizing a fitness function. GAs and evolutionary computation (EC) based on evolutionary principles of Darwinism have been applied to new domains for many years [1-5]; e.g. to test intelligent controllers [6]. Other researchers have utilised GAs to aid financial forecasting [7], co-synthesize hardware-software in embedded systems [8], and recently, they have been applied to optimization problems in the nuclear energy industry [9].

Evolutionary Computing has considerable potential as a simulation tool in the Requirements Engineering (RE) domain, since the process of RE involves incrementally refining requirements to meet users' needs or fitness criteria. Requirements monitoring has tracked the performance of implemented systems against their original requirements so, when deviations were discovered [10], the requirements drift could be corrected by adapting or maintaining the system. However this monitoring operated post implementation, if requirements could be monitored and adapted pre implementation, then an automated solution could be found. Scenarios have been run in simulations of required systems to validate system behaviour, with explanation facilities that illustrate the consequences of system behaviour in tables or animations [11, 12]. However scenario based validation incurs the problem of completeness, .i.e. finding a necessary and sufficient set of scenarios to ensure all requirements have been checked. This paper develops previous research by synthesizing influences from requirements monitoring, and simulation based requirements validation with techniques from evolutionary computing to evolve requirements automatically.

The paper is structured in 4 sections. The design architecture is described in section 2. Section three describes the results of case study in a command and control system. Finally, brief conclusions and future work are presented in section four.

II. SYSTEM ARCHITECTURE

A. Evolutionary Requirements Analysis

Scenarios have a diverse definitions in RE [12]. Nearly all scenarios have requirements implicit within them. Our challenge is to extract the design or high-level requirements from a scenario and then elaborate possible permutations in requirements specification to find an optimal fit with the environment in which the designed system will operate. Translated into evolutionary computing terms we are proposing to take an initial requirements specification as a starting point, and then explore variations within the specification. The EC concepts of crossover and mutation are applied to the initial requirements specification to breed new variants, then the performance of each requirement variant is interpreted and measured against the fitness criteria. Applying evolutionary computing in this manner solves the combinatorial explosion problem in scenario based requirements validation. We can compute a large number of possible variations and then subject them to computational natural selection to discover which variants of the initial specification have survived

To realise this idea necessitates adapting some of the basic tenets of evolutionary computing. In contrast to genetic algorithms there is no executable design (or algorithm), hence we have to interpret the performance of the requirements specifications. This is done by assigning system components expected values for criteria such as usability, reliability, performance effectiveness, etc. Selection criteria are the requirements trade offs that we wish to explore such as cost-reliability.

The domains that we are exploring with this approach are complex; for example, command and control systems involve complex networks of human operators interacting with many different types of equipment. The requirements problem is to

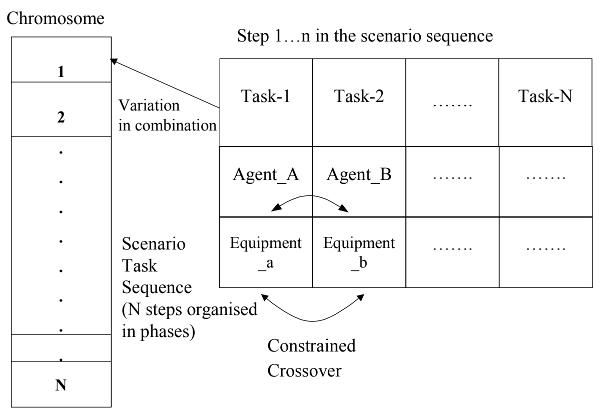

FIGURE 1. STRUCTURE OF THE SCENARIO TASK SEQUENCES, PHENOTYPE.

discover which combination of human agents and computer equipment will be optimal within an environment described in one or more operational scenarios. For instance in a naval command and control system, how will a warship survive scenarios with a range of threats (1-n) with sea state (1-n) and visibility (1 – m) etc. The requirements model is governed by the known constraints and properties of the domain. A generic command and control task model is used to model the scenarios. Each step in the scenario consists of a tuple describing the task, the human operator and the computerised component that supports the human or automates the task. The structure of the phenotype representation is illustrated in figure 1. The first level represents the scenario as a series of steps. The second level describes each scenario step as a combination of an agent, task, and equipment type.

The domain model of agents, equipment types and tasks is represented in Table I.

The tasks are components of a generic command and control cycle. Each phase in a scenario involves one cycle of the command and control task from Monitor/ detect event, Interpret significance, Analyse the situation, Plan the response, Direct personnel how to respond, execute the response. A single scenario will go through several cycles of the command and control task.

The case study models the response of a warship to an air launched missile threat. The scenario consists of five phases with each phase containing 5 task, agent, and equipment tuples apart from phase 2 that has two additional tasks for communication and coordination. Each task in the cycle is carried out by a different human role/ technology combination. In phase one the hostile aircraft was detected and the radar jamming was used as the response. Phase 2 starts when the hostile aircraft launches a missile to attack the ship. To counter the attack, the electronic counter measures are taken to confuse the missile. In phase 3 the missile is still homing on the ship so decoys are launched to divert the missile from its target. The decoys fail to stop the missile, so in phase 4 the decision is taken to launch a defensive missile to destroy the hostile missile. Luckily this is successful and the final phase, reports the mission was completed. The possible combinations of agents and equipment type for scenario phase 1 are illustrated in table II.

TABLE I.
AGENTS, TASKS AND EQUIPMENT TYPES IN THE DOMAIN MODEL

	Symbol	Descriptions
Agent	APC	Air Picture Compiler
	TPC	Tactical Picture Compiler
	TPS	Tactical Picture Supervisor
	PWO	Principle Weapons Officer
	WDB	Weapon Director Blind
Task	MONI	Monitor
	ANAL	Analysis
	PLAN	Plan
	DIRE	Direct
	OPERJ	Operate (execute)
Equipment Type	RAD1-4	Radar type 1 to 4
	CTD	Console Tactical Display
	GPS	Global Positioning System
	NAUTIS	NAUTIS Command system
	JAM 1-4	JAM type 1 to 4

TABLE II
THE SUBSTITUTION TABLE FOR AGENTS AND EQUIPMENT TYPES

Tasks	Agent Substitutions	Equipment Type Substitutions
MONI	APC,SPC,TPC,TPS	RAD1,RAD2,RAD3,RAD4
ANAL	PWO,OOW,CMD,TPS	GPS,CTD,RADIO1-4, NAUTIS
PLAN	CMD,PWO,OOW	CTD,GPS,PD,NAUTIS
DIRE	PWO,CMD,OOW	CTD,GPS,NAUTIS
OPERJ	WDB,WDV,WCCO, EWD	JAM1,JAM2,JAM3,JAM4

Design variants created crossover operators produce different combinations of operators and equipment for each scenario step. The performance of all design variants is assessed to determine which ones pass the fitness criteria reliability of the human agent and equipment when carrying out the task.

B. System Architecture and the Evolutionary Process

The conceptual architecture of the Evolutionary Requirements Analyser is illustrated in figure 2. The stating point is a narrative scenario that is elaborated using the generic command and control task to create a system model describing the human operators and technical equipment used to carry out tasks at each scenario step. We modelled scenarios in a semi-structured format, called Scenario Task Sequences (STSs).

The STSs are coded to facilitate EC crossover and mutation operations in the Chromosome mapper. The mapping mechanism consisted of a lookup table and modulo function to encode and decode between STSs phenotypes and genotypes.

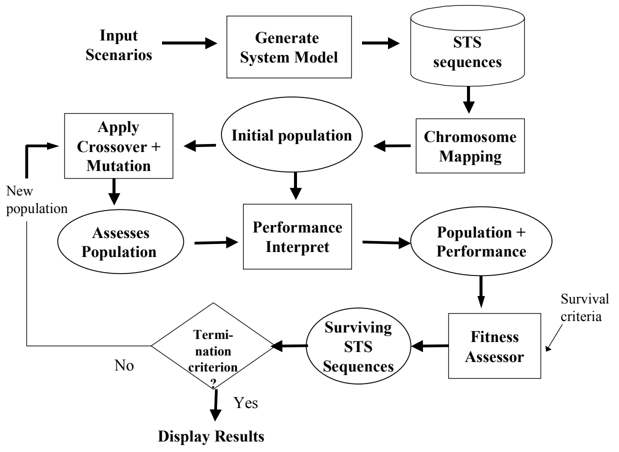

FIGURE 2. ARCHITECTURE AND MODEL OF THE EVOLUTIONARY PROCESS

The system model and chromosome structure was designed in two levels. The top level coded the task in each scenario step, while the second level described the task- agent-equipment type combination within a step. The binary string for the chromosome was 20 bits long, with agent and equipment type assigned 5 bits and the task 10 bits. The initial population was set to 16 and the combination of agent and equipment types were generated randomly within the constraints imposed by the domain (see Table II for legal combination).

Crossover operations were applied to swap agents and equipment types between scenario steps governed by a set of constraints that specified legal operator/equipment combinations in our real world domain. Crossovers were determined by ranking the top performing chromosome and then applying tournament selection to the ranked list. The Crossover policy used a simple "one-cut-point" method and roulette wheel selection[13]. Mutation operators changed agent and equipment types as permitted within the domain model constraints. Mutations were generated by generating random numbers between 0 and 1, using a selection rate set at P= 0.005 to determine the mutation and then mapping from the mutated genotype to an agent/equipment phenotype (see table 2) via a modulo function.

The control algorithm for the EC process differed from normal GAs applications because of the need for an interpreter to assess performance of the STS models.

Algorithm for the control process

```
Begin
  Map STS (Phenotype)  to Genotype;
  Gen=0;
  Random create (Population(Gen 0));
 While (termination criterion not met) {
     Interpret Performance (Gen n)
     Select Phenotypes Reliability>.= x
     Mutate Genotypes (Gen n)
     Crossover Genotypes (Gen n)}
  Map Genotype to Phenotype
End of process
```

The performance of each STS sequence was determined by the performance interpreter that modeled the expected influence of training, task complexity, operational time, quality and utility on system reliability hence this was our equivalent of running a genetic algorithm to obtain a performance value. Performance for each scenario step was calculated by formulae (1).

$$E_s(w,p) = V_w \times V_p \qquad (1)$$

where, $V_w = [w_{Agent-\mathrm{Re}l}, .., w_{Task-Comp}, .., w_{ET-Qua}, w_{ET-\mathrm{Re}l}, ...]$

$$V_p = \begin{bmatrix} p_{RAgent} \\ \cdot \\ \cdot \\ p_{CTask} \\ \cdot \\ \cdot \\ p_{QET} \\ p_{RET} \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

'V_w' is the weighting vector for each component property, while 'V_p' is the vector contained the value of properties that were derived from [14]. Input values of reliability for each agent role and equipment type were extracted from look-up tables for the relevant scenario step (e.g. for the Monitor task , the APC agent and RADAR equipment types). The input values for each property of the relevant agent and equipment type were weighted according to estimated contribution that the human agents and equipment made to each task, for instance in Monitor tasks, reliable equipment is more important, whereas in Planning a well trained human agent is vital. The weighting factors that increased or decreased the performance of agents and equipment summed to 1.0 to represent the proportional influence of the variables. The values calculated for each scenario step were summed to yield a combined error value (E). Error values for all scenario steps were then averaged to give an overall value for the whole scenario, as detailed in equation (2).

$$F(E_{STS}) = 1 - E_{STS} \qquad (2)$$

Where, $E_{STS} = \frac{\sum_{s=1}^{N} E_s}{N}$, N: *number of tasks in a STS*

Global values for the agent and equipment type reliabilities were calculated from different property settings. Each agent and equipment type property could be set to high, medium, or low levels as preconditions in dependency rules that produced a range of values for agent and equipment reliabilities, e.g.

IF Utility = High AND Quality = Low THEN
Equipment Type reliability factor = 0.04

A pair wise comparison of the 4 variables that influenced the equipment types produced 81 possible combinations (e.g. RADAR 1, GPS etc); similarly comparison of the 2 agent variables produced 9 combinations for each agent role (PWO, APC, etc). A single equipment type and agent value is chosen by the requirements analyst to represent the agent and technology properties to be tested for each scenario. The fitness of chromosomes was modified with the selected f_{Agent} and f_{ET} values to increase or decrease STS reliability, as shown in equation (3).

$$F(E_{STS}) = F(E_{STS}) * (1 - f_{Agent}) * (1 - f_{ET}) \qquad (3)$$

This produced a final performance value for the chromosome that represented the particular design variant. Performances of all the STSs were evaluated by the Fitness assessor which selected chromosomes with reliability rates above a minimum level set by the user. These chromosomes were passed onto the next generation, and in the case that too severe a fitness level had been selected, the top 8 performing chromosomes were taken. The next generation was produced by elitist selection to ensure the best chromosomes were passed onto next generation. No chromosomes were duplicated for the next generation to prevent the domination of super chromosome. The generation run was set to 100. Optimal results were collected throughout the run to discover the STSs with maximum fitness.

III. RESULTS

To demonstrate the effectiveness our system, we applied the evolutionary requirements analyzer to a naval command and control system using the missile threat scenario and combinations of human agents, tasks and equipment described in Table 1 and 2. The results for STSs are illustrated in figure 3 and 4. Scenario phases 1, 3, 4 and 5 had the same number of tasks in their STSs and showed a similar evolutionary process, hence, only phase 1 is illustrated for reference. The process converged in less than 10 generations, probably because the search space was constrained so the EC process could easily select the optimal solution. Figure 4 shows the result of phase 2 which showed slower convergence and higher oscillation during the evolutionary process. Since phase 2 contained 7 tasks, the EC process needed more generations to find an optimal solution. Another observation in phase 2 was the effect of mutation rate. With a binary string length of 20 * 7 =140 bits and a mutation rate of P_m = 0.005, allowed approximately 11 opportunities (with the N= 16 initial population) in every generation. That is relatively high value might have caused the oscillation shown in the phase 2 (see figure 4). The implication is that mutation rates should be applied carefully when using long string chromosome to stabilise the evolutionary process.

The optimal STS compared with the original seed population are shown in table III. Substitutions in agents and equipment types are underlined.

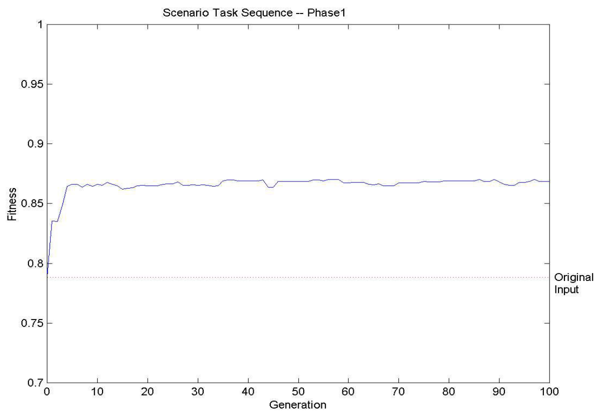

FIGURE 3 THE RESULTS OF PHASE 1

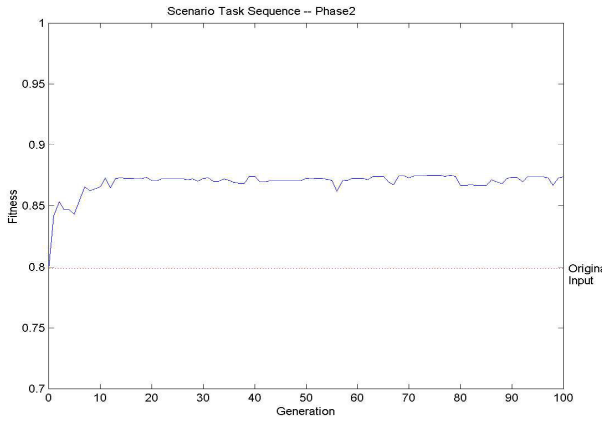

FIGURE 4 THE RESULTS OF PHASE 2

TABLE III
FITNESS COMPARISON TABLE

	STS – Phase 1	Fitness
Origin	<APC,MONI,RAD1> <TPS,ANAL,CTD> <PWO,PLAN,NAUTIS> <PWO,DIRE,GPS> <EWD,OPERJ,JAM1>	0.78
Optimum	<TPC,MONI,RAD3> <PWO,ANAL,NAUTIS> <PWO,PLAN,CTD> <PWO,DIRE,NAUTIS> <WDB,OPERJ,JAM3>	0.87

Fitness improved about 10% compared to the original scenario and, as expected, the agent/equipment type combinations have changed in the optimal solution. The tasks, “MONI” , “ANAL”, and “OPERJ” changed both the agent and equipment type. The tasks, “PLAN” and “DIRE” only changed the equipment type. PWO was assigned three tasks in the optimal STS, reflecting the high training and reliability values of this agent. However the increased workload may exceed a human operators capacity to undertake this role, so this illustrates the need for further rules to model performance degradation under high workload that would counter balance the tendency of the current process to optimize task allocation to the most able agents.

In this paper we have only reported the results from analysis of one scenario. When other scenarios are run with the same system model many of the agents, task and equipment types as shown in Tables 1 and 2 will be the same, although some additions are necessary to describe different steps. Each run optimises the system model for that particular scenario, so we then have to synthesise a common (required) model of system components. Currently this is achieved manually; however, in the future we will investigate adapting and transferring optimal genotypes between scenarios to automatically evolve a solution across several scenarios.

IV. CONCLUSION

This preliminary study has demonstrated a novel application of Evolutionary computing to requirements engineering. This has presented interesting challenges because we are attempting to evolve specifications and discover how well they fit within their environment, in contrast to the traditional EC approach of evolving an executable artefact for optimal performance. There are several avenues for future exploration. First we will incorporate the influencing factor permutations into the evolutionary process so we can seed the process with a more varied population of agents and equipment types with differing capabilities. Next we intend to elaborate the fitness criteria, so requirements trade offs can be explored, for instance between performance time and reliability. This theme can be taken further by introducing environmental variables into fitness assessment, so we can determine if design (x) performs well under environmental conditions (i..j). This will entail further work on the dependency rules.

The selection process we used optimised evolutionary forces to favor the best performing chromosomes following the infect genes approach of Travares et al who restricted crossover and mutation to the more promising genes [15]. Our approach is related to the use of GAs to generate test scripts for usability evaluation by Kasik and George [16], and one extension to our system could employ GAs to produce new scenarios as test data for our system models. In conclusion this work has demonstrated a novel application of evolutionary computing and produced a new architecture to address the problem of evolving requirements specifications and selecting components in socio technical system designs.

A. *References*

[1] D. Fogel, Evolutionary Computation: toward a new philosophy of machine intelligence. Piscataway, NJ: IEEE Press, 1995.

[2] J. Holland, Adaptation in Natural and Artificial Systems: University of Michigan Press, 1975.

[3] M. Gen and C. Runwei, Genetic Algorithms & Engineering Design: John Wiley & Sons, Inc., 1996.

[4] T. Bäck, U. Hammel, and H.-P. Schwefel, "Evolutionary Computation: Comments on the History and Current State.," *IEEE transactions on Evolutionary Computation*, vol. 1, 1997.

[5] D. B. Fogel, "What is Evolutionary Computation?," in *IEEE Spectrum, February 2000*, 2000, pp. 26 - 32.

[6] A. C. Schultz, J. J. Grefenstette, and K. a. De Jong, "Applying Genetic Algorithms to the Testing of Intelligent Controllers," presented at The Workshop on APplying Machine Learning in Practice at IMLC-95, 1995.

[7] S. Chiraphadhanakul, P. Dangprasert, and V. Avatchanakorn, "Genetic Forecasting Algorithm with Financial Applications," presented at Proceedings of the 1997 IASTED International Conference on Intelligent Information, Grand Bahama Island, BAHAMAS, 1997.

[8] R. P. Dick and N. K. Jha, "MOGAC: A Multiobjective Genetic Algorithm for the Co-Synthesis of Hardware-Software Embedded Systems," presented at Proceedings of the 1997 International Conference on Computer-Aided Design (ICCAD '97), San Jose, CA, 1997.

[9] V. G. Toshinsky, H. Sekimoto, and G. I. Toshinsky, "A method to improve multiobjective genetic algorithm optimization of a self-fuel-providing LMFBR by niche induction among nondominated solutions," *Annals of Nuclear Energy 27, 2000*, vol. 27, pp. 397 - 410, 2000.

[10] S. Fickas and M. S. Feather, "Requirements Monitoring in Dynamic Environments," presented at IEEE second international symposium on Requirements Engineering., York, England, 1995.

[11] C. L. Heitmeyer, James Kirby, and B. Labaw., "Tools for Formal Specification, Verification, and Validation of Requirements," presented at Proceedings of 12th Annual Conference on Computer Assurance (COMPASS '97),, Gaithersburg, MD., 1997.

[12] C. Rolland, B. C. Achur, C. Cauvet, J. Ralyte, A. Sutcliffe, N. Maiden, M. Jarke, P. Haumer, K. Pohl, E. Dubois, and P. Heymans, "A Proposal for a scenario classification framework.," *Requirements Engineering Journal*, vol. 3, pp. 23 - 47, 1998.

[13] Z. Michalewicz, Genetic Algorithm + Data Structure = Evolution Programs, 2nd ed: Springer-Verlag, New York, 1994, 1994.

[14] A. G. Sutcliffe, "Requirements Engineering for Complex Collaborative Systems," presented at RE01, 5th IEEE international Symposium on Requirements Engineering,, Toronto, Canada, 2001.

[15] R. Tavares, A. Teofilo, P. Silva, and A. C. Rosa, "Infected Genes Evolutionary Algorithm," presented at Proceedings of the 1999 ACM symposium on Applied Computing, San Antonio, TX, 1999.

[16] D. J. Kasik and H. G. George, "Toward automatic generation of novice user test scripts," presented at Conference proceedings on Human factors in computing systems, 1996.