EFFECTIVE CONCURRENT ENGINEERING WITH THE USAGE OF GENETIC ALGORITHMS FOR SOFTWARE DEVELOPMENT

D.Sundar[1], Dr.K.Alagarsamy[2]

[1] Assistant Professor, Thiagarajar School of Management, Madurai, India
[2] Associate Professor, Computer Center, Madurai Kamaraj University, Madurai, India

ABSTRACT

Software specifications are useful to the organizations to enforce a consistent approach in designing, implementing and the maintenance of the software. There are numerous principles employed for the assurance of the quality of the software. Concurrent engineering with Genetic algorithms have been adopted for the software development. The results obtained are appreciable and detailed discussion is done.

KEYWORDS:

MOGA, Concurrent Engineering etc.

1 INTRODUCTION

Concurrent engineering is a business approach which replaces the conventional product development method with one in which tasks are done in parallel and there is an early consideration for every aspect of a product's development process. This policy focuses on the resources of the enterprise to be dispersed in the devising and the implementation process to assure an efficient software development process.

In today's business world, corporations must be able to react to the changing market needs rapidly, effectively, and responsively. They must be able to reduce their time to market and adapt to the changing environments. Decisions must be made quickly and they must be made right the first time out. Enterprises could not offer the process to be repeated which incurs more time and cost to deliver the product to the channel. Concurrent engineering is being the alternative has emerged as a way of bringing rapid solutions to product design and development process.

Concurrent engineering is unquestionably the gesture of the future for new software development for all enterprises in spite of of their size, complexity or product range. In order to be outstanding, enterprises must modify their deliverables and software development cycle to be able to whole varied processes concurrently. This new model will add advantage to the enterprise, although it will require a large amount of refinement in its implementation. Concurrent Engineering is to be properly assessed before the deployment. This is because concurrent engineering is a process that must be analyzed and accustomed for unremitting enhancements of business process.

DOI : 10.5121/ijsea.2012.3507

Concurrent engineering is not a one size fits all solution to a firm's development processes. This continuous improvement cycle consists of planning, implementing, reviewing, and revising. This must be regularly investigated and the process must be reviewed in the due course of implementation to enhance the process of concurrent engineering.

2 BASIC PRINCIPLES OF CONCURRENT ENGINEERING

The concurrent process model can be represented schematically as a series of major technical activities, tasks, and their associated states. For example, the development activity defined for the spiral model is consummated by initiation the following tasks: prototyping and/or analysis modeling, requirements specification, and design [1].

The earliest design decisions in product development have significant impact on the operational and support (O&S) aspects of the software product. These support considerations can best be handled concurrently to the planning, requirements definition, and high level design activities [2].

The advantageous things that the concurrent engineering can invite are more even but the process is difficult for deployment. The payback will be not only enjoyed by the enterprise deploying but also the users, stakeholders and the end consumers. They reap the benefit of using the concurrent engineering by reduction of the cost, time etc. The cross functional team also enjoys the benefit, but not depended on the applications alone. [3].

Some of the Concurrent Engineering (CE) multidisciplinary team structures include [4] Functional Team, Light weight Team, Heavyweight team, autonomy team, Collocated Autonomy Team and Virtual team.

The effective new business product development process can decrease the development lead times which also enhances the upstream and downstream processes which are to be considered at the conceptual development phase [5]. This approach is typically described as Concurrent Engineering. Hence, CE in an organization signifies the ability of the organization to embrace product development as a series of overlapping stages, which provides customer satisfaction and also the right price by delivering products on time. This is effectively accomplished by employing numerous engineering tools and system techniques during the project management of design product development.

3 PROBLEM FORMULATION

Firms looking for bloodthirsty advantage to amplify market share, turnover, and expansion have bowed to concurrent development to pace with the introduction of novel products and strike their rivals to market. Concurrent engineering is a complex problem in that a large number of considerations in the software development have to be brought to bear during the design stage. The problems are likely to encompass constraints of widely varying complexity. Previous approaches have concentrated on finding good solutions to simplified problems or, alternatively, to finding a feasible solution which may not be close to optimal. The researcher presents an approach to concurrent engineering problems that extends the multi objective genetic algorithm approach to handle the complexity that is inherent in a typical concurrent engineering problem. This poses some severe research challenges.

Software engineering problems grow in complexity and so too grow the models necessary to solve these problems. This increased complexity translates directly to a greater number of required activities, longer model evaluation times, and slower design development. One way of

overcoming this obstacle is through the optimization of the sequence of routines in the development activities.

Organizations involved in concurrent software development not only experience the dynamics common to single projects, but also face interactions between different releases of their product: they share resources among different stages of different projects, including customer support, they have a common code-base and architecture that carries problems across releases, they use the same capabilities, and their market success in early releases impacts their resources in later ones.

Software phases which involve the various resources for the allocation must have and to meet some pre defined purpose. Input data for schedule planning and optimization are a set of routines and their durations, description of resource constraints and a set of priority relations.

i. Let R = (1,2,…n} a set of routines,
   dr (where r Є R) is the duration of the rth routine, dr > = 0 for every r.
ii. Priority relations in a set R as a set of pairs
   P={ (i,j) where i must be done before j}
iii. Set of available resources for the project development L = {1,….l}.
iv. Set of human resources H = { 1, 2,…h} and each h Є H is associated with a set of S = { s1,s2,….sh} the skill set of each human resource.

So the fitness function can be drafted based on the following objective functions

1. Minimize (L ,H)

subject to

2. $Rk \leq Ri\text{-}Ri \quad i = 1,\ldots,n,\ k \in Pi$
3. $Ri \geq 0 \qquad i = 1,\ldots,n$

The objective function

a) Minimizes the resources (both L and H) required for the process n.
b) Imposes the precedence relations between process and constraints
c) Forces the resource to be non-negative.

4 PROPOSED METHODOLOGY

Genetic algorithms (GAs) are powerful, general purpose adaptive search techniques which have been used successfully in a variety of learning systems. In the standard formulation, GAs maintain a set of alternative knowledge structures for the task to be learned, and improved knowledge structures are formed through a combination of competition and knowledge sharing among the alternative knowledge structures. J. David Schaffer extended the GA paradigm by allowing multidimensional feedback concerning the performance of the alternative structures. The modified GA is shown to solve a multiclass pattern discrimination task which could not be solved by the unmodified GA.

Multi Objective evolutionary algorithms using the properties like non non-dominated sorting and distribution is criticized for the complexity of the algorithm, non elitist property and not specifying the parameter for the distribution.

Kalyanmoy Deb suggested a non-dominated sorting based multi-objective evolutionary algorithm (we call it the Non-dominated Sorting GA-II or NSGA-II) which alleviates all the above three difficulties. In this algorithm computational complexity is addressed. The problem of the

selection operator is also addressed. Simulation results on a number of difficult test problems show that the proposed NSGA-II, in most problems, is able to find much better spread of solutions and better convergence near the true Pareto-optimal front compared to the PAES and SPEA two other elitist multi-objective EAs which pay special attention towards creating a diverse Pareto-optimal front. Moreover, we modify the definition of dominance in order to solve constrained multi-objective problems efficiently. Because of NSGA-II's low computational requirements, the elitist approach, the parameter-less niching approach, and simple constraint-handling strategy, NSGA-II should find increasing applications in the coming years.

The presence of multiple objectives in a problem, in principle, gives rise to a set of optimal solutions (largely known as Pareto-optimal solution), instead of single optimal solution. This type of problem is known as multi-objective optimization problem (MOP). In general, a MOP has been solved using weighted sums or decision-making schemes. The problem is to look for the alternative Pareto-optimal front. An alternative way is to look for the Pareto-optimal front.Many evolutionary algorithms (EAs) like genetic algorithm (GA) have been suggested to solve a MOP, and hence termed as multi-objective evolutionary algorithms (MOEAs). Amar Kishor, 2004 presented an application of NSGA-II in order to solve a multi-objective series system reliability optimization problem. Here, the conflicting objectives such as the system reliability maximization and system cost minimization have been considered [16] – [15].

5 EXPERIMENTAL ENVIRONMENT

The data for the study is collected from a software firm located in Madurai. The pilot survey is made by the interaction with the Project Head and the Human Resource Manager of the firm. The existing allocation is done manually and the skill of the employees is measured in the numeric scale rated for fitting into the project.

The chosen problem has ten resources allocated for a set of five jobs. The expected cost using MOGA is given in table 5.1. The expected time for development using MOGA is given in table 5.2. The expected efficiency using MOGA is given in table 5.3.

TABLE 5.1

EXPECTED COST BASED ON THE RESOURCE USAGE USING MOGA (IN THOUSANDS)

Number of Job(i)/ resources (j)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	300	299	314	306	336	337	325	314	303	319
2	312	309	290	319	337	345	336	336	325	329
3	322	323	336	337	349	389	336	323	338	322
4	292	314	336	367	314	349	337	303	334	390
5	314	336	294	323	336	317	305	337	296	336

TABLE 5.2

TIME BASED ON THE PRIORITIZATION OF THE CONSTRAINTS USING MOGA (IN MAN DAYS)

Number of Job(i)/ Constraints (j)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	42	53	68	74	49	88	101	79	110	121
2	79	105	66	78	109	70	69	55	111	67
3	74	110	59	44	105	111	44	86	59	83
4	53	99	77	69	50	100	79	110	66	87
5	49	99	109	66	99	77	111	77	91	77

TABLE 5.3

EXPECTED EFFICIENCY BASED ON MOGA

Number of the Jobs	**Efficiency using MOGA**
1	6
2	9
3	5
4	9
5	7

The environmental set up for the execution based on the multi objective genetic algorithm is summarized in Table 5.4

TABLE 5.4

ENVIRONMENTAL PARAMETERS FOR MOGA

Parameters	**Values**
Population size	100
Crossover rate (C)	0.8
Mutation rate (M)	0.1
Stopping criteria	100 generations

Based on the above environment the concurrent engineering is applied and the objectives of the minimization of resources, imposing the precedence of the constraints and the minimization of the development time are enforced. The results are discussed below.

6 RESULTS AND DISCUSSIONS

The experiments are carried out and the results have been depicted in terms of graphs. In the above discussed environment the following results are obtained.

Table 6.5 gives the results obtained from the optimized resource usage based on cost through the application of MOGA with concurrent engineering.

TABLE 6.5
COST BASED ON THE CONCURRENT ENGINEERING
(IN THOUSANDS)

Number of Job(i)/ resources (j)	1	2	3	4	5	6	7	8	9	10
1	298.7	296.3	312.97	303.21	334.21	335.28	322.89	311.78	300.23	316.87
2	310.934	307.78	287.4	316.32	334.76	342.98	333.72	333.98	323.1	326.7
3	320.1	321.02	334.02	335.02	347.02	387.02	334.02	321.02	336.02	320.71
4	289.54	311.91	333.67	365.78	311.91	346.98	334.91	300.02	331.28	386.9
5	312.9	335.14	293.14	322.98	335.14	316.14	304.14	336.14	295.12	335.14

TABLE 6.6

PERCENTAGE OF IMPROVEMENT IN COST

Number of Job(i)/ resources (j)	1	2	3	4	5	6	7	8	9	10
1	0.433333	0.90301	0.328025	0.911765	0.532738	0.510386	0.649231	0.707006	0.914191	0.667712
2	0.341667	0.394822	0.896552	0.840125	0.664688	0.585507	0.678571	0.60119	0.584615	0.699088
3	0.590062	0.613003	0.589286	0.587537	0.567335	0.508997	0.589286	0.613003	0.585799	0.400621
4	0.842466	0.665605	0.693452	0.332425	0.665605	0.578797	0.620178	0.983498	0.814371	0.794872
5	0.350318	0.255952	0.292517	0.006192	0.255952	0.271293	0.281967	0.255193	0.297297	0.255952
Mean	0.511569	0.566479	0.559966	0.535609	0.537264	0.490996	0.563847	0.631978	0.639255	0.563649

In fig 6.1 the improvement between the existing approach and the proposed approach is calculated based on the cost. The cost associated with the both the methods have been compared and the proposed method outperforms the existing method.

FIG 6.1 : THE IMPROVEMENT BETWEEN THE EXISTING APPROACH AND THE PROPOSED APPROACH

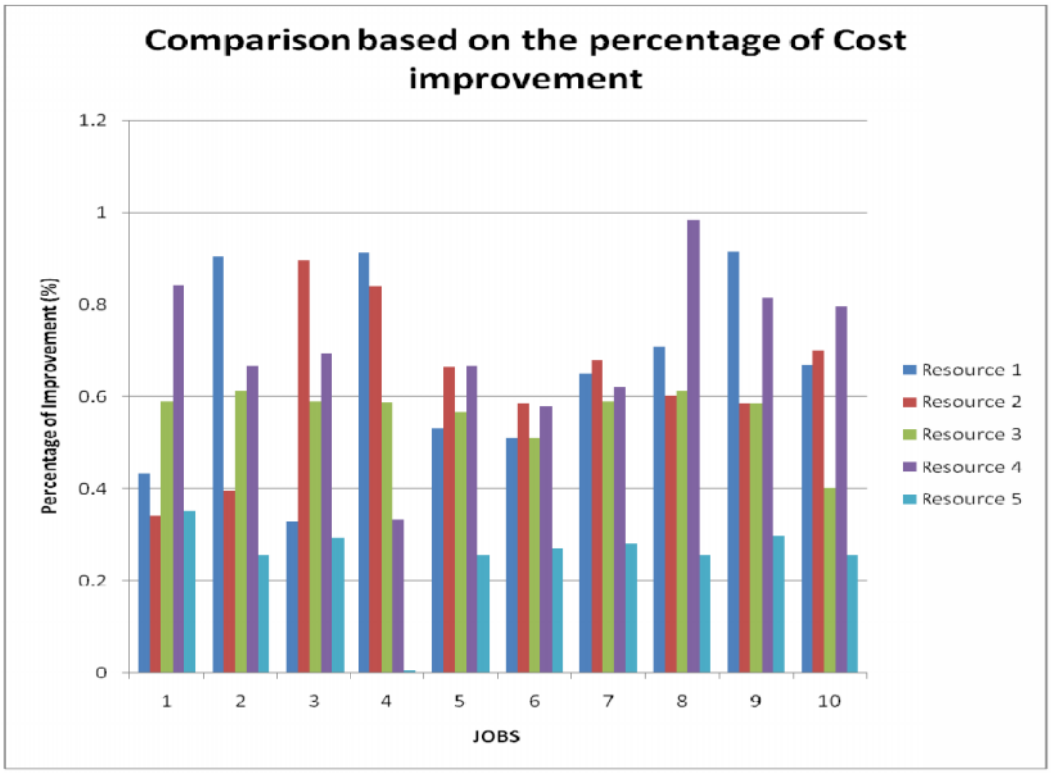


Table 6.7 gives the results obtained from the optimized software development time based on cost through the application of MOGA with concurrent engineering.

TABLE 6.7
DEVELOPMENT TIME BASED ON THE CONCURRENT ENGINEERING (IN MAN DAYS)

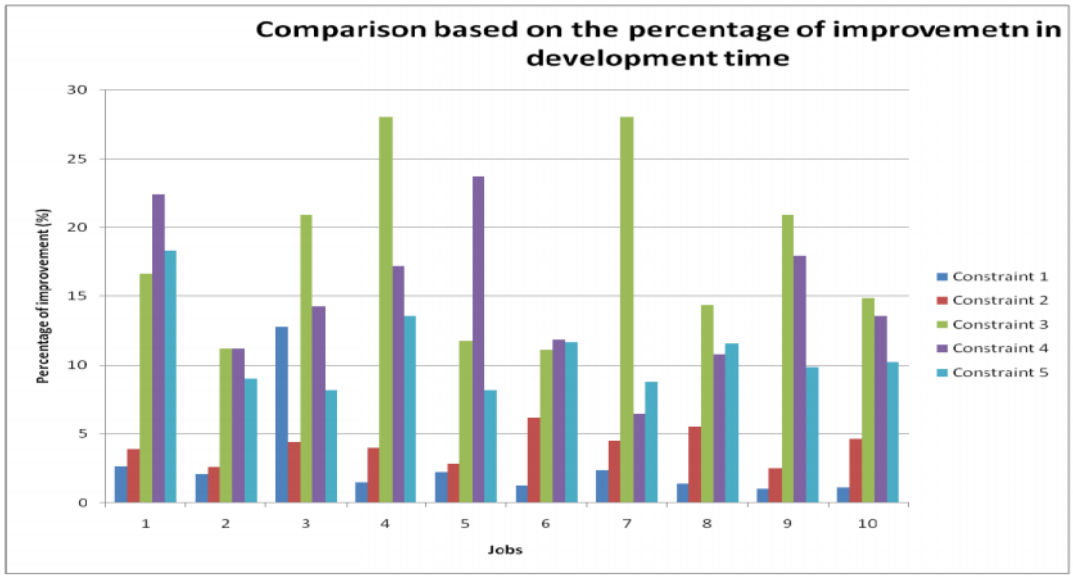
Number of Job(i)/ Constraints (j)	1	2	3	4	5	6	7	8	9	10
1	40.9	51.9	59.3	72.9	47.9	86.9	98.6	77.9	108.9	119.67
2	75.91	102.3	63.1	74.91	105.91	65.7	65.91	51.98	108.2	63.91
3	61.66	97.66	46.66	31.66	92.66	98.66	31.66	73.66	46.66	70.66
4	41.13	87.92	66.02	57.13	38.13	88.13	73.9	98.13	54.13	75.19
5	40.02	90.045	100.09	57.04	90.87	68.02	101.23	68.09	82.02	69.13

TABLE 6.8 PERCENTAGE OF IMPROVEMENT IN DEVELOPMENT TIME

Number of Job(i)/ Constraints (j)	1	2	3	4	5	6	7	8	9	10
1	2.619048	2.075472	12.79412	1.486486	2.244898	1.25	2.376238	1.392405	1	1.099174
2	3.911392	2.571429	4.393939	3.961538	2.834862	6.142857	4.478261	5.490909	2.522523	4.61194
3	16.67568	11.21818	20.91525	28.04545	11.75238	11.11712	28.04545	14.34884	20.91525	14.86747
4	22.39623	11.19192	14.25974	17.2029	23.74	11.87	6.455696	10.79091	17.98485	13.57471
5	18.32653	9.045455	8.174312	13.57576	8.212121	11.66234	8.801802	11.57143	9.868132	10.22078
Mean	12.78577	7.220491	12.10747	12.85443	9.756853	8.408462	10.03149	8.718898	10.45815	8.874815

In the fig 6.2 the improvement between the existing approach and the proposed approach is calculated based on the time of development under the given constraints using the concurrent engineering. The proposed method outperforms the existing method.

FIG 6.2 THE IMPROVEMENT BETWEEN THE EXISTING APPROACH AND THE PROPOSED APPROACH



In table 6.9 the deviation measure has been computed and the deviation indexes prove that the performance of the proposed system changes on the case basis. The other influencing factor is that the various constraints are considered.

TABLE 6.9
STANDARD DEVIATION OF THE RESULTS OBTAINED IN THE PROPOSED APPROACH

Std. Dev	Jobs									
	1	2	3	4	5	6	7	8	9	10
Cost	0.210145	0.250568	0.253621	0.37366	0.167906	0.128073	0.161035	0.260994	0.239287	0.225772
Time	8.948591	4.559859	6.27792	10.70643	8.749859	4.64444	10.34701	5.201533	8.927221	5.883138

7 CONCLUSION

A novel approach has been proposed in the paper to trounce the existing approaches using concurrent engineering process. The problem for the Human resource management is based on their skills, number of resource availability and the cost attached to the person for the multiple modules has been considered. The priorities and the sequence of routines are also concentrated. MOGA obtained the best Pareto optimal solutions when compared to the manual assignment method. A comparative analysis table and graph are provided at the end of this paper. Form this we can conclude that an Optimized approach for the Improvement of Capability Maturity Model Integration (CMMI) in Human Resource Management using Multi Objective Genetic Algorithms can be used. In future we can extend this technique for other issues unsolved by CMMI.

8 REFERENCES

[1] http://www.mymanagementguide.com/project-constraints-and-project-assumptions-planning-for-project-success.

[2] http://wiki.answers.com/Q/What_is_the_concurrent_development_model.

[3] Keene, K.C., Keene.S.J, Concurrent engineering aspects of software development, Proceedings. Of Software Reliability Engineering, 1992.

[4] http://best.berkeley.edu/~pps/pps/concurrent.html.

[5] R. Addo-Tenkorang, Concurrent Engineering (CE): A Review Literature Report, Proceedings of the World Congress on Engineering and Computer Science 2011 Vol II, WCECS 2011, October 19-21, 2011, San Francisco, USA.

[6] Tennant, C., and Roberts, P., (2000), 'A faster way to create better quality products'.International Journal of Project Management, Vol. 19, pp. 353–362.

[7] Abdullah Konak, David W. Coit, Alice E. Smith, Multi-Objective Optimization Using Genetic Algorithms: A Tutorial, ie.rutgers.edu/resource/research_paper/paper_05-008.pdf.

[8] Coello, C.A.C., A comprehensive survey of evolutionary-based multi objective optimization techniques, Knowledge and Information Systems 1(3) (1999) 269-308.

[9] Coello, C.A.C. An updated survey of evolutionary multiobjective optimization techniques: state of the art and future trends. In Proceedings of the 1999. Congress on Evolutionary Computation-CEC99, 6-9 July 1999. 1999. Washington, DC, USA: IEEE.

[10] Coello, C.A.C., An updated survey of GA-based multi objective optimization techniques, ACM Computing Surveys 32(2) (2000) 109-143.

[11] Fonseca, C.M. and Fleming, P.J. Genetic algorithms for multi objective optimization: formulation, discussion and generalization. in Proceedings of ICGA-93: Fifth International Conference on Genetic Algorithms, 17-22 July 1993. 1993. Urbana- Champaign, IL, USA: Morgan Kaufmann.

[12] Jensen, M.T., Reducing the run-time complexity of multi objective EAs: The NSGA-II and other algorithms, IEEE Transactions on Evolutionary Computation 7(5) (2003) 503- 515.

[13] Xiujuan, L. and Zhongke, S., Overview of multi-objective optimization methods, Journal of Systems Engineering and Electronics 15(2) (2004) 142-146.

[14] Zitzler, E., Deb, K., and Thiele, L., Comparison of multi objective evolutionary algorithms: empirical results, Evolutionary Computation 8(2) 173-195.

[15]Zitzler, E. and Thiele, L., Multi objective evolutionary algorithms: a comparative case study and the strength Pareto approach, IEEE Transactions on Evolutionary Computation 3(4) (1999) 257-271.