8 EXPERIMENT: QUALITATIVE ELEGANCE

8.1 Background

The role of elegance in early lifecycle software design is one that would appear to have not stimulated a great deal of research interest. However, Gelernter (1998) examines the notion of 'machine beauty' and suggests it can be found *"…in a happy marriage of simplicity and power – power meaning the ability to accomplish a wide range of tasks, get a lot done"*. Indeed, Gelernter discusses the *"the aesthetics of computer science"* and points to the recursive 'quicksort' algorithm as an example of a beautifully designed sorting algorithm that is simple in design yet powerful in performance. It is interesting to speculate that Gelernter's notion of 'machine beauty' appears to be in agreement with the principle of Occam's Razor with its emphasis on parsimony, economy and succinctness, although Gelernter does not specifically refer to Occam's Razor in his book. Buschmann and Henney (2010a, 2010b) do however explore such notions with respect to the design of software architecture. Indeed, Buschmann and Henney begin by posing the question: *"what are the five top properties that make a software design both effective and elegant?"* and go on to suggest that software design economy, visibility, spacing, symmetry and emergence provide *"a perspective on software architecture, a value system that can broadly guide architects' design decisions"*. The authors prudently suggest that it is necessary to balance such considerations however, since the dogged pursuit of one consideration may negatively impact another. For example, too much economy in a design might reduce the overall size, but visibility is lost.

Is elegance in software design significant? Certainly in the 1990s, authors such as Parnas (1996) bemoaned a lack of elegance in the software of the era. Parnas pondered why software with a consistent style and simple, organised components was so hard to find, or in other words, *"why software jewels are so rare"*. However, design elegance emerged as a crucial factor in the rise of the software design patterns community. Gabriel (1996) discusses patterns of software design and cites Harbison (1992): *"There is a pleasure in creating well-written, understanding software. There is a satisfaction in finding a program structure that tames the complexity of an application. We enjoy seeing our algorithms expressed clearly and expressively. We also profit from our clearly written programs, for they are much more likely to be correct and maintainable than obscure ones"*. The notion of design elegance helping to tame complexity and enhancing maintainability are pursued further by Gabriel who

notes that symmetry among design components is highly useful in defining the dependencies and hence granularity of large scale software designs. Such ideas are directly incorporated by Gamma *et al.* (1995) in their seminal design patterns catalogue wherein *"patterns solve specific design problems and make object-oriented designs more flexible, elegant and ultimately reusable"*. These ideas are later explored by Wirfs-Brock (2007) who ponders the beauty of software design and code. Wirfs-Brock claims that brevity can contribute to code beauty through clarity of purpose and expressive use of the programming language, but only within an elegant design context. Wirfs-Brock also revisits the work of Gabriel, but argues against beauty as an overarching goal in itself. Rather, she suggests that elegance is significant in software design with respect to the development and maintenance of software designs in the face of inevitable change (as discussed, for example, by Boehm and Beck, 2010), since elegant designs *"preserve and make evident the designer's intent"*. Wirfs-Brock goes on to invoke Gabriel's notion of software 'habitability', in which software engineers coming to a software design later in its life *"…understand its construction and intentions and change it comfortably and confidently"*. This notion is consistent with the simple conclusion of Tractinsky *et al.* (2000) that *"what is beautiful is useable"*.

In the empirical investigation described in the previous chapter, it is interesting to note that by the end of the design episode supported by the interactive framework, the participants were freely providing suggestions for enhancing and extending the interactive framework. Indeed, one of the suggestions made was the capability to record a measure what the participants termed the 'quality of appearance' of the software design, possibly as a star rating. It is also interesting to speculate on possible causative factors that could have brought this suggestion to the fore. For example, the use of quantitative measures such as external design coupling in the empirical study occasionally steers the search to designs that are quantitatively superior to the manually produced design. However, the appearance of these designs might suggest to the participants that they had not been arrived at by human designers. Reflecting on the inherent nature of the quantitative fitness measures, it is apparent that high values for class cohesion are easier to achieve in small classes, but harder to achieve in large classes. Thus a software design of high fitness as measured quantitatively by cohesion and coupling metrics might be comprised of one very large class (with many attributes and methods) together with a number of small classes (with just one attribute and one method). However, the class designs produced manually by designers for the example

design problem in the empirical investigation tend to possess a more even distribution of attributes and methods among classes. In short, designs produced by quantitative measures alone may not be elegant. It is interesting to note that this is consistent with the work of Brown *et al.* (1998) who report 'anti-patterns'. Brown *et al.* describe an anti-pattern as *"a commonly occurring solution to a problem that generates decidedly negative circumstances"*, and cite examples of anti-patterns together with refactoring suggestions to rectify the anti-pattern. Intriguingly, Brown *et al.* report one anti-pattern named the "Blob" class (also known as the "God" class). In the "Blob" class anti-pattern, a class design is dominated by one monopolising controller class among many small classes – a very inelegant design. Refactoring suggestions to rectify the "Blob" anti-pattern involve moving attributes and methods from the huge class to other smaller classes to achieve a more even distribution of attributes and methods among the classes of the design and promote elegance. It is interesting to speculate that in the empirical investigation, although quantitative fitness functions steer the search to the discovery of software designs of superior quantitative fitness, these solutions may or may not exhibit design elegance, and even occasionally reveal anti-patterns. This strongly suggests that for effective search, both quantitative measures and qualitative evaluation of design 'fitness' are required to steer the search. To achieve this, the nature of the interaction between the human designer and any computational design support is crucial.

8.2 Proposed Approach

Given that the evidence presented in the previous section suggests that elegance is important in early lifecycle software design, in what ways might quantitative fitness measures and qualitative designer evaluations be effectively integrated for interactive search and exploration? Furthermore, within this interactive quantitative and qualitative search of software designs, how might interaction fatigue be reduced through machine-learning techniques such as dynamic multi-objective search and reward-based learning? To address these questions, the proposed approach builds upon the findings of the empirical study reported in the previous chapter to comprise four components. Each component is described as follows.

8.2.1 Quantitative Elegance Measures

Firstly, it is proposed that elegance in software designs is related to the symmetry and evenness of distribution among elements in a design (albeit in some way that is designer

and design context dependent). Therefore, evenness of distribution is quantified though measurements of the distribution of attributes and methods among classes. Four novel quantitative measures of software design elegance are proposed as follows:

- *Elegance 1* is the standard deviation of the numbers of attributes and methods among the classes of a design and is calculated as follows. Firstly, the average number of attributes per class in a design is calculated together with the standard deviation. Secondly, the average number of methods per class in a design is calculated together with the standard deviation. Elegance 1 is calculated as the average of the two standard deviations. The notion here is that the lower the value for elegance one, the more symmetrical the appearance of attributes and methods among the classes in the design as a whole.
- *Elegance 2* is the standard deviation of external couples among the classes of a design and is calculated as follows. For each class in the design, the number of external couples is recorded. The average number of external couples per class is calculated, together with the standard deviation. Elegance 2 is this standard deviation. The notion here is that the lower the value the elegance 2, the more even the distribution of external couples among design classes.
- *Elegance 3* is the standard deviation of 'internal' uses within the classes of a design and is calculated as follows. For each class in the design, the number of internal uses is recorded. (An internal use occurs when a method in a class 'uses' an attribute in the same class). The average number of internal uses per class is calculated, together with the standard deviation. Elegance 3 is this standard deviation. The notion here is that the lower the value the elegance 3, the more even the distribution of internal uses among design classes.
- *Elegance 4* is standard deviation of the ratio of attributes to methods within the classes of a design and is calculated as follows. For each class in the design, the ratio of attributes to methods is calculated. The average for all ratios is calculated together with the standard deviation. Elegance 4 is this standard deviation. The notion here is that the lower the value of elegance 4, the more symmetrical the appearance of attributes and methods in individual classes of the design.

It is evident that all the above elegance measures are minimisation functions. However, for the purposes of interactive search and exploration, it is also interesting to note that

during colourful visualisation of the early lifecycle software designs, outward indications of elegance measures 1, 2 and 4 are visible but elegance measure 3 is not.

8.2.2 Role of Quantitative Elegance Measures in Reducing User Fatigue

Secondly, it is proposed to use the quantitative elegance measures as one of the tactics to reduce user fatigue. Indeed, user fatigue has been well recognised as an implementation issue with interactive evolutionary computation (IEC) systems (e.g. Tagaki, 2001, Caleb-Solly and Smith, 2007, Shackelford, 2007). According to Shackelford (2007), IEC systems can be perceived by the user as time-consuming and boring, while Caleb-Solly and Smith (2007) point out that human evaluation might show non-linearity of focus over time. To address these issues, Tagaki (2001) suggests a number of techniques, including:

- reducing the number of fitness values selected,
- displaying multiple solutions to the user in a pre-sorted order,
- reducing the number of solutions displayed,
- reducing the number of iterations, and
- embedding user knowledge in the application.

With this in mind, it is noted that as reported in the empirical study of the previous chapter, colourful visualisations of early lifecycle software designs in UML format are engaging but semantically rich. Indeed, there may be many human considerations of elegance for the designer to ponder when evaluating a single software design visualisation. Because of this, it is proposed to:

- visualise a single design solution from the population to the user, and
- invite the designer to provide a 1 to 5 ‘elegance’ star rating (as suggested from the empirical investigation reported in the previous chapter).

Pre-sorting and ranking multiple solutions for display to the user has been considered (e.g. Shackelford and Corne, 2001, Machwe and Parmee, 2009) but rejected as the risk of information overload is high, possibly leading to user boredom and non-linearity of focus over time. It is important to also note that although only one design solution is presented to the designer, the population size remains at the relatively high numbers reported in the previous chapters in order to retain effective search and exploration and preserve population diversity. Therefore, to reduce user fatigue, the single most elegant solution from the population is presented to the designer from the population for

elegance evaluation. Specifically, one of the four quantitative elegance measures is selected at random and this measure is used to choose the single most elegant software design solution from the population for visualisation. It is proposed that this mechanism provides 'natural' looking UML class design visualisations for the designer, rather than the so-called 'machine-produced' class designs reported in the empirical study. Furthermore, it provides a means to investigate what elegance measures, if any, are favoured by the designer.

8.2.3 Elegance Agent and Dynamic Multi-objective Search

Thirdly, in an additional attempt to reduce user fatigue, it is proposed that an autonomous *Elegance Agent* monitors the designer elegance evaluation for each of four elegance measures. Regarding the designer elegance evaluation as feedback or 'reward', the Elegance Agent calculates the mean reward for each of the four elegance measures as the design episode progresses. The Localised Search Agent then uses this mean reward to dynamically update the proportionate selection weights thus producing an interactive, dynamic and multi-objective search. In this way, as the Elegance Agent learns designer elegance intentions, dynamic multi-objective search is steered to reflect those designer elegance intentions.

Building on the results of previous chapters, localised search, performed by a Localised Search Agent, has been chosen to investigate dynamic, multi-objective search. A non-dominated sorting approach to multi-objective search has been considered but rejected due to the difficulties of dynamically adjusting the relative weights of the five fitness functions. Perhaps a more straightforward approach is provided by weighted sum methods, which scalarise a set of objectives into a single objective by multiplying each objective with a user-supplied weight. The issue of what values to provide for the weights is crucial, although Parmee *et al.*, (2002b) and Machwe and Parmee (2009) address this with some success in interactive design situations by using qualitative preference and aesthetic information respectively to quantify the weights. However, setting up an appropriate weight vector also depends upon the scaling of each objective function to normalise the objectives, which is difficult to achieve for the four elegance measures, given the diversity and scale of the three software design problems used in this investigation. A different approach that does not require scaling of objectives has been suggested by Schaffer (1995), which he called the vector evaluated genetic algorithm (VEGA). In VEGA, the population is divided

into equally sized subpopulations for each objective function to be optimised. Each subpopulation is then assigned a fitness value based on a different objective function. After each solution is assigned a fitness value, the selection operator, restricted among solutions of each subpopulation, is applied until the mating pool for the subpopulation is filled. In this manner, restricting the selection operator only within a subpopulation emphasises good solutions corresponding to that particular objective function. Indeed, since no two solutions are compared for different objective functions, disparity in the ranges of different objective functions does not create difficulty either. Thus Schaffer's vector evaluated genetic algorithm (VEGA) has inspired the localised search used in this investigation, although a number of enhancements are proposed to reflect the interactive and dynamic context of localised search. Indeed, it is crucial that the localised search is computationally efficient (so as to not detrimentally impact designer interaction and bring on user fatigue) and yet retain population diversity. Because of this, separate subpopulations are not maintained. Rather, a single population of software design solutions and a proportionate selection operator is proposed. Building on results of previous chapters, the selection operator used is binary tournament selection. Self-adaptation is also used as the mutation probability mechanism to maintain population diversity. However, the tournament selection operator uses the appropriate fitness function (i.e. external coupling or one of the four elegance metrics) for tournament comparison in proportion to the dynamic elegance 'reward' received from the designer as the evolutionary search progresses.

Let the interactive designer elegance evaluation be regarded as 'reward', *r*, for the four elegance metrics, $r_{e_1}, r_{e_2}, r_{e_3}$ and r_{e_4} respectively. The Elegance Agent computes a mean reward for each elegance measure based on the sequence of designer interactive evaluations during the design episode as follows

$$\bar{r}_{e_1} = \frac{1}{n}\sum_{k=1}^{n} r_{e_1,k} \tag{8.1}$$

where *n* is the number of interactive rewards provided by the designer. The mean reward values for the other elegance values $\bar{r}_{e_2}$, $\bar{r}_{e_3}$ and $\bar{r}_{e_4}$ are similarly defined. Let the selection weights of the five quantitative measures be w_c, the weight for external design coupling, and $w_{e_1}, w_{e_2}, w_{e_3}, w_{e_4}$ the weights for the four elegance measures respectively. The scale of the selection weight for each elegance measure is equal i.e.

$$0.0 \leq w_{e_1}, w_{e_2}, w_{e_3}, w_{e_4} \leq 0.2$$

The selection weight for external coupling is calculated as follows:

$$w_c = 1.0 - (w_{e_1} + w_{e_2} + w_{e_3} + w_{e_4}) \quad (8.2)$$

Thus when the selection weights for each elegance metrics are at maximum values (i.e. 0.2), the selection weights for all fitness functions are equal.

At the beginning of search prior to designer interaction, the selection weight for each elegance measure is zero and so external coupling is used as the sole selection metric over the entire search population. However, as search progresses, the designer provides elegance evaluations of designs for an elegance measure chosen at random, from which the Elegance Agent updates the mean reward for the chosen elegance measure. The Elegance Agent then communicates the updated mean reward to the Localised Search Agent which updates the selection weights for each elegance measure as follows:

$$w_{e_i} = \bar{r}_{e_i} . c \quad (8.3)$$

A value of 0.04 is used for the constant, *c*, as this effectively maps the scale of the 1 to 5 reward star rating to the upper value of the elegance weightings of 0.2. At each designer interaction, the weighting for external coupling is also updated as in equation (2). In this manner, the dynamic multi-objective search emphasises quantitative external coupling at the start of search, but as designer interactions increasingly contribute to search, the selection weightings of elegance measures increase. Indeed, should the designer reward one elegance measure above all others, search is increasingly steered to design solutions reflecting this elegance measure. The interaction selection weighting update mechanism is summarised as follows:

1) Randomly select an elegance measure, e_i, from elegance 1..4
2) Select most elegant design from population using e_i
3) Present visualisation of elegant software design
4) Obtain designer evaluation in range 1 to 5 (star rating)
5) Update mean reward $\bar{r}_{e_i}$ for e_i
6) Update selection weighting w_{e_i} based on mean reward $\bar{r}_{e_i}$
7) Update selection weighting w_c

A flow chart illustrating the overall proposed dynamic multi-objective interactive search approach is shown in figure 8.1.

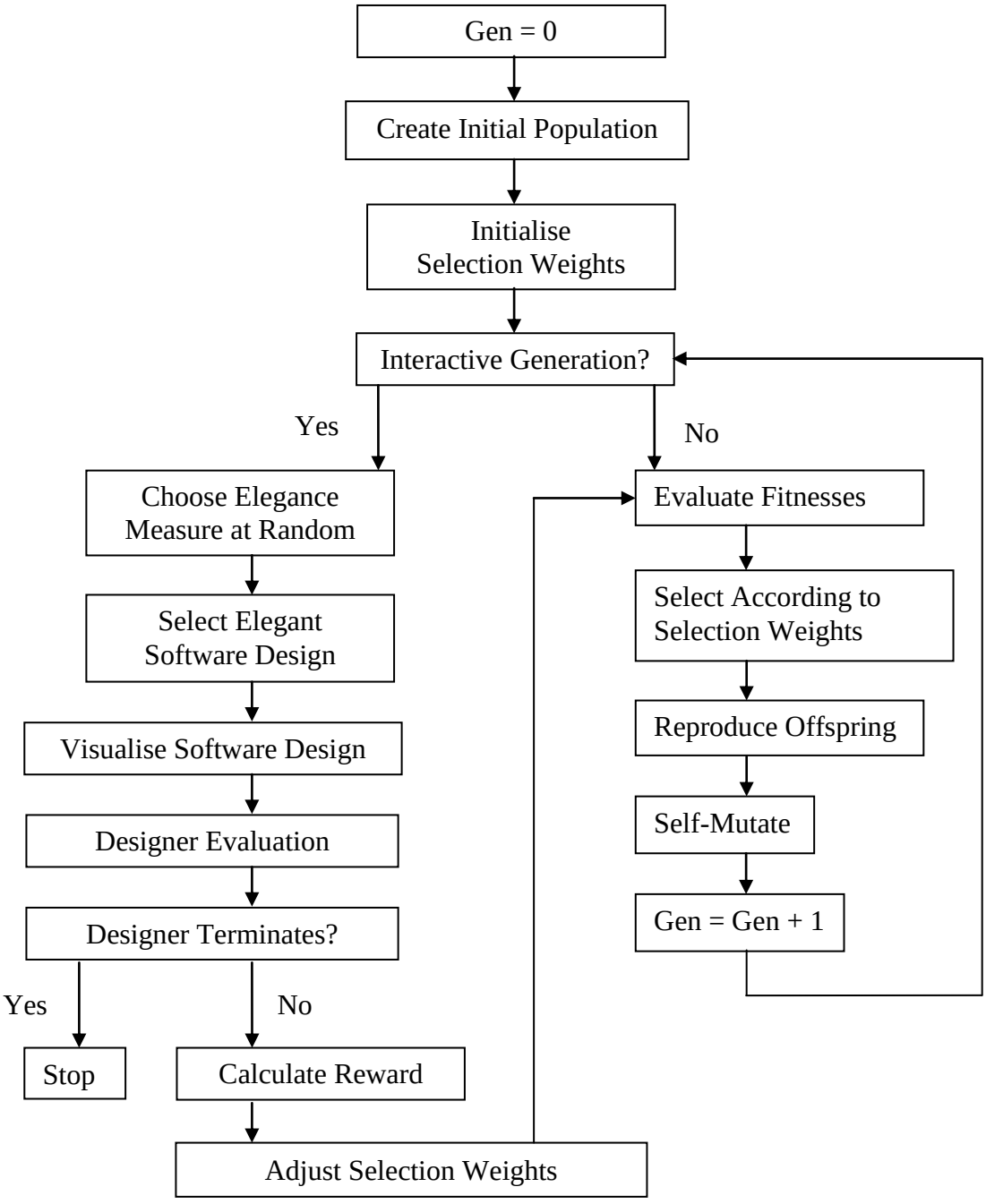

Figure 8.1. Flow Chart of Dynamic Multi-objective Interactive Evolutionary Search

8.2.4 Dynamic Fitness Proportionate Interactive Interval

Fourthly, again relating to user fatigue, it is crucial to strike a balance between quantitative external design coupling and elegance measures. Typical responses to

potential user fatigue have been to reduce population size and reduce the number of solutions displayed at every generation to the user for scoring or ranking (e.g. Tagaki, 2001, Caleb-Solly and Smith, 2007, Machwe and Parmee, 2009). However, in early lifecycle software design for example, if the designer evaluates a software design at every generation, it is possible that the overall average population external coupling fitness might not progress sufficiently, since user boredom may bring about early halting of search. On the other hand, if the designer evaluates a software design at too large a generational interval, there is insufficient interaction, the elegance measures do not influence the direction of search, and so 'natural', elegant designs may not be arrived at. It is therefore proposed that the generational interval of each designer interaction be dynamic and fitness proportionate.

Early lifecycle software design external coupling fitness ranges from 0.0 to 1.0, and is a minimisation function. It is proposed that the interactive interval of search in terms of the number of generations is proportional to the square of external coupling fitness. The expectation is that the better the coupling fitness, the lower the interactive interval thus enabling more frequent designer interaction. A dynamic interaction interval is achieved by the following calculation:

$$\text{Interaction Interval (number of generations)} = \text{coupling fitness}^2 \,.\, ic \qquad (8.4)$$

where *ic* is an interval constant. Example values of interactive interval in generations are given in table 8.1; values giving an interactive interval of 1 or less are shaded. Hence when the average population external coupling is inferior, the generational interaction interval is relatively high. However, as the average population external coupling becomes increasingly superior, the generational interactive interval becomes lower to increasingly allow designer elegance evaluations to influence the direction of search.

8.2.5 Overview of Proposed Approach

Taken in the round, the above proposed approach builds upon but differs from that employed by Brintrup *et al.* (2007) and Brintrup *et al.* (2008) in manufacturing plant layout and ergonomic chair design respectively. In both studies, Brintrup *et al.* report multi-objective search combining quantitative and qualitative fitness measures but the balance between quantitative and qualitative measures is static. The above proposed approach also builds upon the dynamic approach reported by Machwe and Parmee

Table 8.1. Interaction Intervals in Generations for various Interval Constants

Coupling Fitness	Interval Constant								
	1	5	10	15	20	25	30	35	40
0.9	0.81	4.05	8.10	12.15	16.20	20.25	24.30	28.35	32.40
0.8	0.64	3.20	6.40	9.60	12.80	16.00	19.20	22.40	25.60
0.7	0.49	2.45	4.90	7.35	9.80	12.25	14.70	17.15	19.60
0.6	0.36	1.80	3.60	5.40	7.20	9.00	10.80	12.60	14.40
0.5	0.25	1.25	2.50	3.75	5.00	6.25	7.50	8.75	10.00
0.4	0.16	0.80	1.60	2.40	3.20	4.00	4.80	5.60	6.40
0.3	0.09	0.45	0.80	1.35	1.80	2.25	2.70	3.15	3.60
0.2	0.04	0.20	0.40	0.60	0.80	1.00	1.20	1.40	1.60
0.1	0.01	0.05	0.10	0.125	0.20	0.25	0.30	0.35	0.40

(2006a, 2006b, 2009) for beam bridge design and urban furniture design. Although Machwe and Parmee report a dynamic quantitative and qualitative multi-objective search approach, they use case-based reasoning and clustering to present multiple solutions for the user to rank; the generational interactive interval is also fixed. While the findings of Brintrup *et al.* and Machwe and Parmee appear promising, it is perhaps not surprising that a different, novel approach is necessary for early lifecycle software design. Although the underlying multi-objective search and exploration techniques show some commonality at an abstract level, the specifics of the different design domains, with their specific representations and associated genetic operators, demand an appropriate and natural interactive approach for each design domain.

8.3 Methodology

Seven software development professionals with experience of early lifecycle software design have participated in trials with the interactive framework using the approach proposed in the previous section for dynamic localised search. Relevant information concerning the seven software professional is given in table 8.2. The total experience of software development of the participants amounts to 133 years. Participant 1 is the

Table 8.2. Trial Participant Information

Participant	Gender	Profession	Years Experience
1	male	Lecturer	20
2	male	Undergraduate Student	3
3	male	Lecturer	30
4	male	Postgraduate Student	25
5	female	Lecturer	10
6	female	Lecturer	20
7	female	Lecturer	25

author of this thesis. Each participant interacts with the interactive framework for early lifecycle software design, using the Cinema Booking System, the Graduate Development Program and Select Cruises as vehicles for investigation. For each software design problem, interval constant values of 1, 5, 10, 15, 20, 25, 30, 35 and 40 are trialled and each interactive design episode is allowed to proceed until the participant decides to halt.

Prior to interacting with the framework, an explanation of each example software design problem is provided to each participant. In addition, the colourful visual designer interface of the interface framework is described. The nature of an interactive design episode is explained to the participant emphasising the design episode as an interactive session of search, exploration and design discovery. However, it is also explained to the participants that the software design episode may or may not be productive in terms of discovering useful and interactive early lifecycle software designs. It is emphasised to participants that they may halt the episode at any point they desire. This might be because that the participant considers that a satisfactory design episode has been conducted in terms of discovering useful and interesting software designs; however it may also be because the participant has become overwhelmed with interaction fatigue. In formulating this approach to trials, it seems logical to expect that more fruitful design episodes might contain a higher number of interactions as the designer becomes satisfied and rewarded with the design discoveries. It would also seem logical to expect that less fruitful design episodes involve a lower number of interactions should user fatigue become predominant. Lastly, it is also important to note that the participant is not informed of which elegance measure has been chosen at

random to select a software design for visual inspection; the participant simply performs their qualitative evaluation of perceived design ‘elegance’.

Values of the interval constant are chosen at random, and then the participants engage in an interactive design episode facilitated by the interactive software design framework. The following information is recorded for each episode:

- example design problem,
- interval constant,
- number of interactions until designer halts, and
- number of interactive framework evolutionary generations at designer halting.

At each evolutionary generation, the interactive framework records:

- external coupling, and
- elegance measures 1, 2, 3 and 4.

At each interaction, the interactive framework records:

- designer qualitative evaluation of design elegance (value ranges from one star to five stars),
- updated mean reward for each elegance measure, and
- updated selection weights for external coupling and each elegance measure.

Lastly, each participant is invited to provide any comments on their overall human experience of the trial. Such comments might include any satisfying aspects, any aspects that generated user fatigue, and any suggestions for enhancement of the interactive framework of overall human experience.

As part of the methodology of investigating the nature of the elegance measures, baseline elegance fitness values will be obtained for each example software design problem in the absence of any designer interaction to observe how elegance values are affected by evolutionary search with external coupling alone.

Building upon findings of previous investigations into dynamic parameter control, interactive trials used localised searches with a population size of 100 individual design solutions, proportionate binary tournament selection and self-adaptation with respect to mutation probability. All local searches have been implemented in the Java programming language and run on a standard Microsoft Windows office desktop PC. Results obtained are reported in the following section.

8.4 Results

Screen shots of example early lifecycle software designs illustrating what might be considered elegant and inelegant designs for the Cinema Booking System, Graduate Development Program and Select Cruises are shown in appendix D.

Results of baseline elegance investigations are presented firstly. Results of investigations into designer interaction and the dynamic interactive interval are presented next, followed by results relating to external coupling and then designer evaluation and reward. After that, results of elegance investigations and dynamic multi-objective search are revealed while lastly, participant comment on the human experience of the interactive framework is presented.

8.4.1 Baseline Elegance Investigations

Localised search has firstly been investigated without human designer interaction to determine what elegance characteristics, if any, might be observed in the absence of human qualitative evaluation of early lifecycle software design elegance. Such investigations act as a baseline for comparison with localised search performed with human designer interaction. Localised search has therefore been executed over 50 runs, to a generation when no further improvement in quantitative elegance is apparent. External coupling is used as the sole fitness objective without human designer interaction. Best, average and worst elegance values for elegance measures 1, 2, 3 and 4 have been recorded. Figures 8.2, 8.3, 8.4 and 8.5 show the averaged results of 50 runs for the Cinema Booking System (CBS), Graduate Development Program (GDP) and Select Cruises (SC) software design problems respectively.

From examination of figures 8.2, 8.3 and 8.4, it is difficult to elucidate what relationship exists, if any, between the quantitative elegance metrics and external coupling as search evolves (apart from loss of diversity as the population converges). Results for elegance metrics 1, 2 and 4 reveal some degree of improvement in elegance as evolution proceeds, but improvements are rarely neither linear nor monotonic. Indeed, results for elegance metrics 1 and 2 reveal a distinct worsening of elegance over initial generations before some later improvement can been seen. The elegance characteristics of elegance 3 results appear dissimilar to the other elegance metrics. If

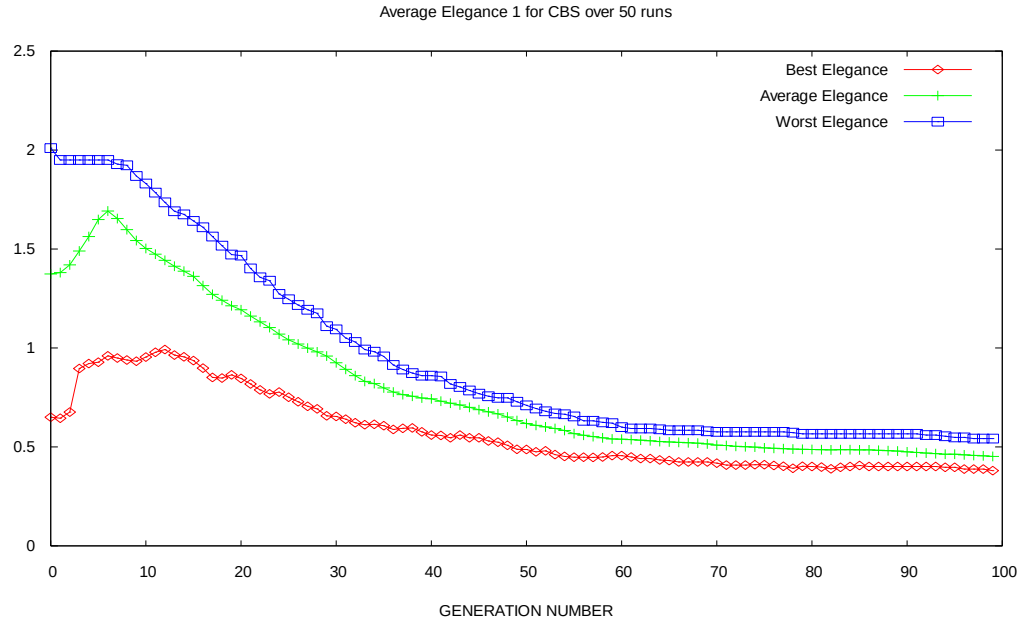

Figure 8.2. Best, Average and Worst Elegance 1 Values for Cinema Booking System

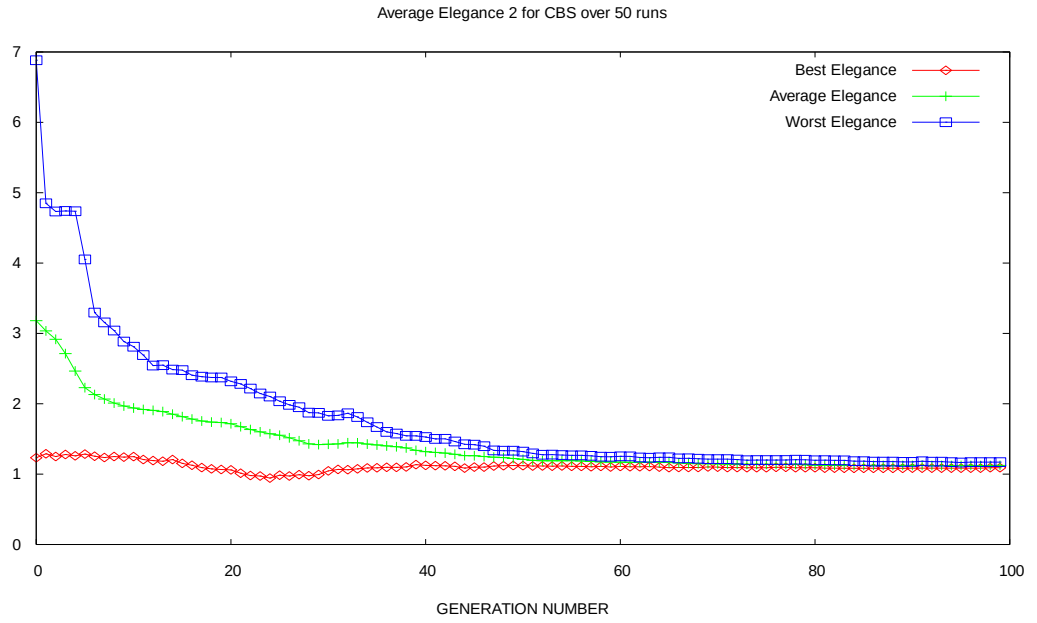

Figure 8.3. Best, Average and Worst Elegance 2 Values for Cinema Booking System

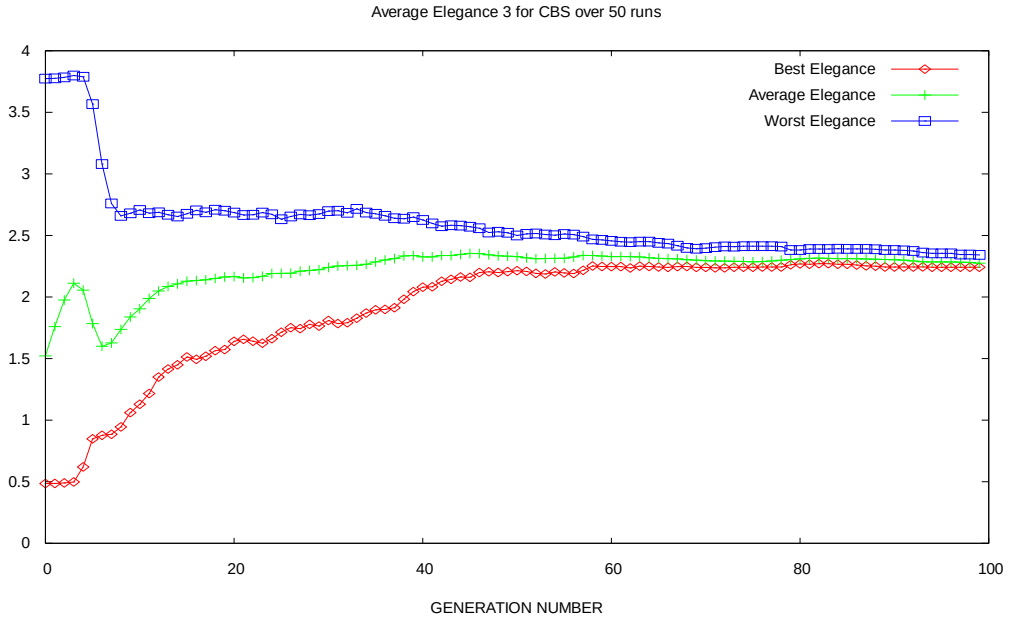

Figure 8.4. Best, Average and Worst Elegance 3 Values for Cinema Booking System

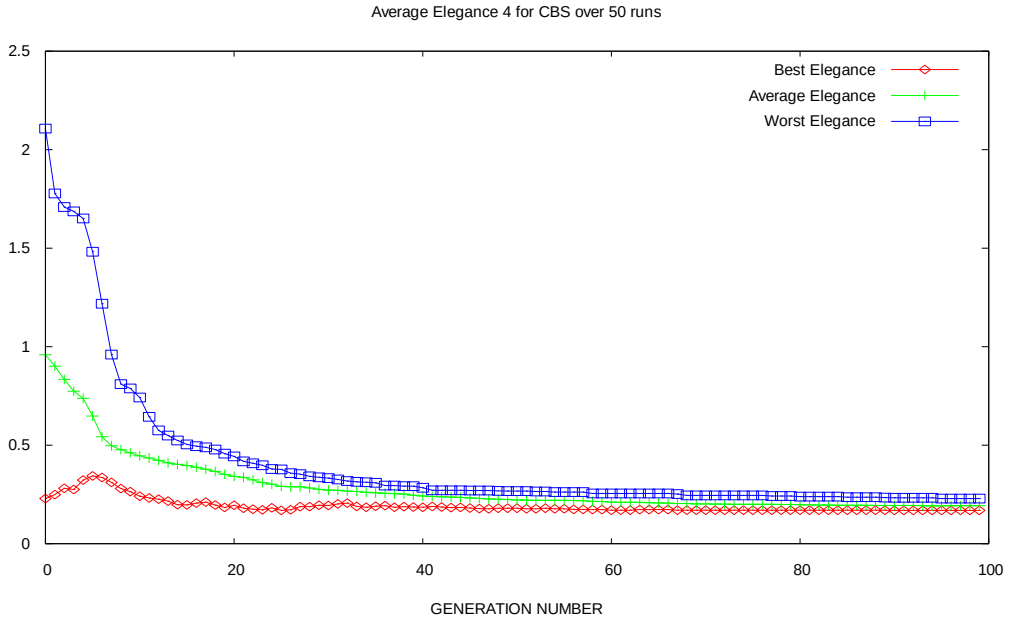

Figure 8.5. Best, Average and Worst Elegance 4 Values for Cinema Booking System

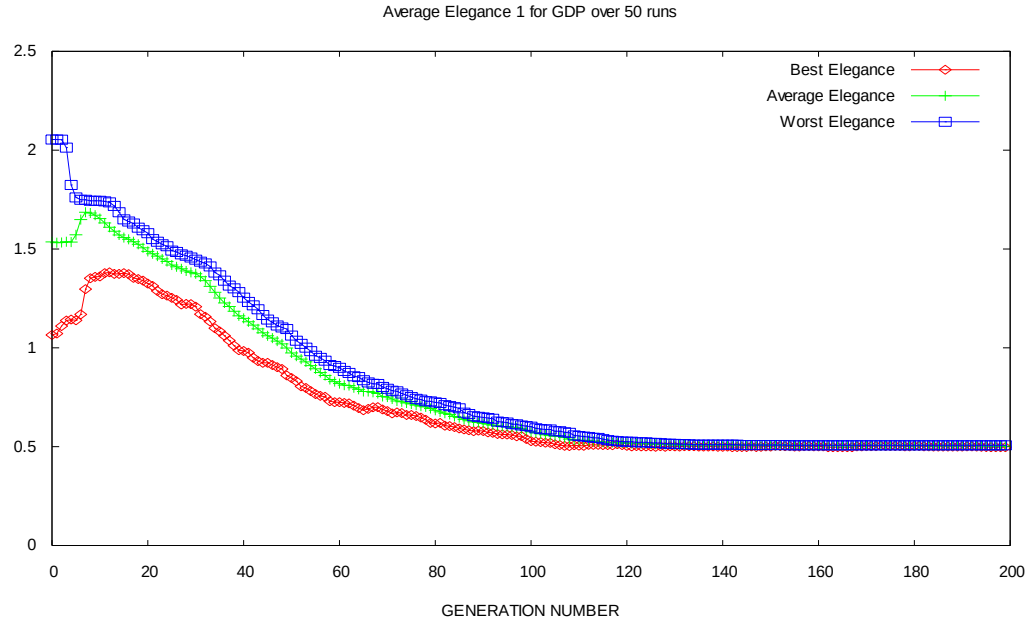

Figure 8.6. Best, Average and Worst Elegance 1 Values for GDP

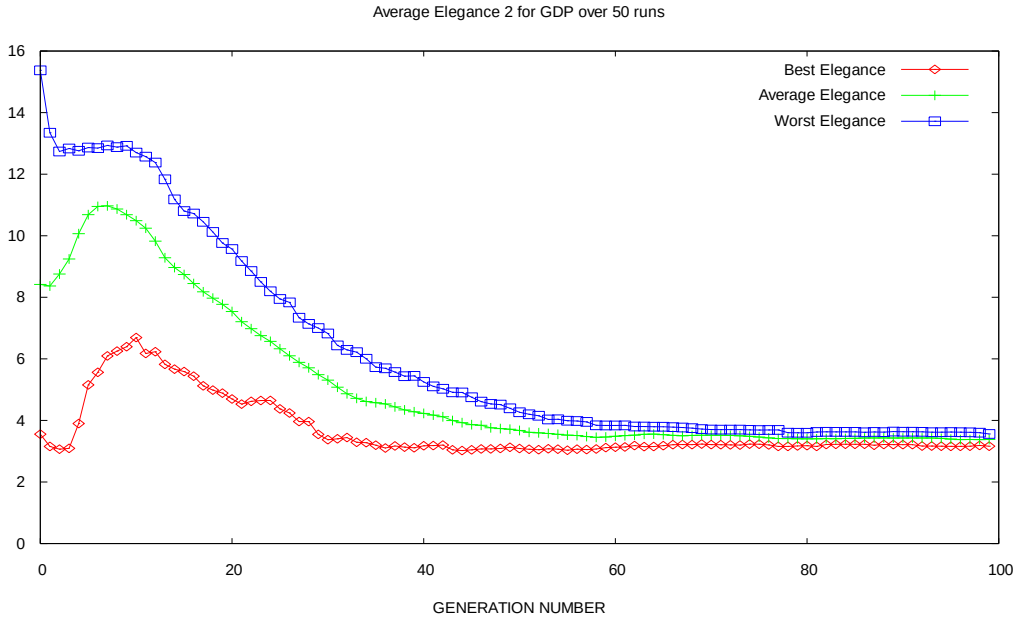

Figure 8.7. Best, Average and Worst Elegance 2 Values for Cinema Booking System

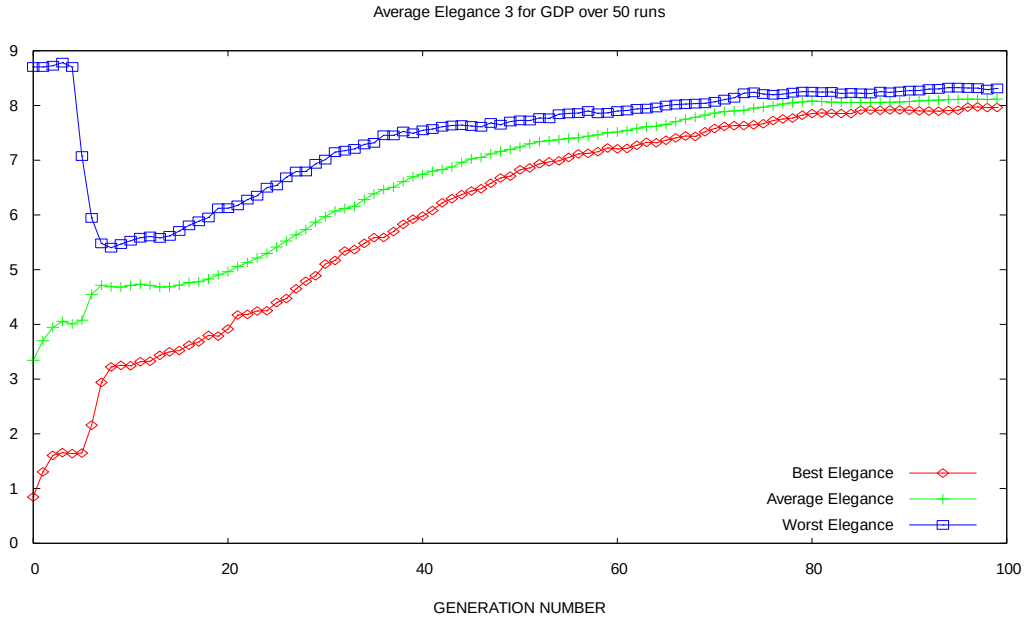

Figure 8.8. Best, Average and Worst Elegance 3 Values for GDP

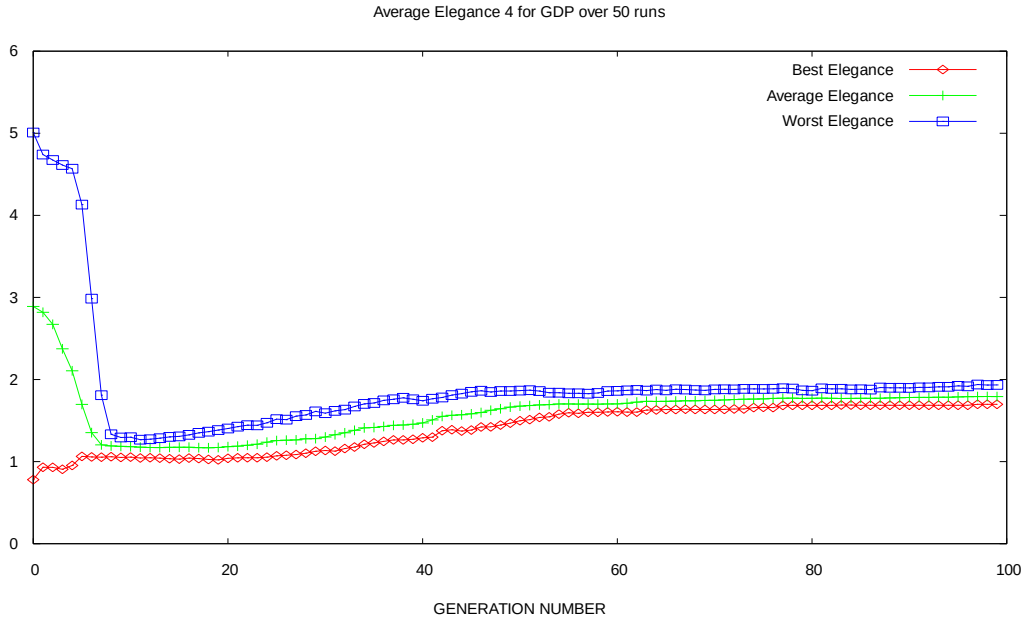

Figure 8.9. Best, Average and Worst Elegance 4 Values for GDP

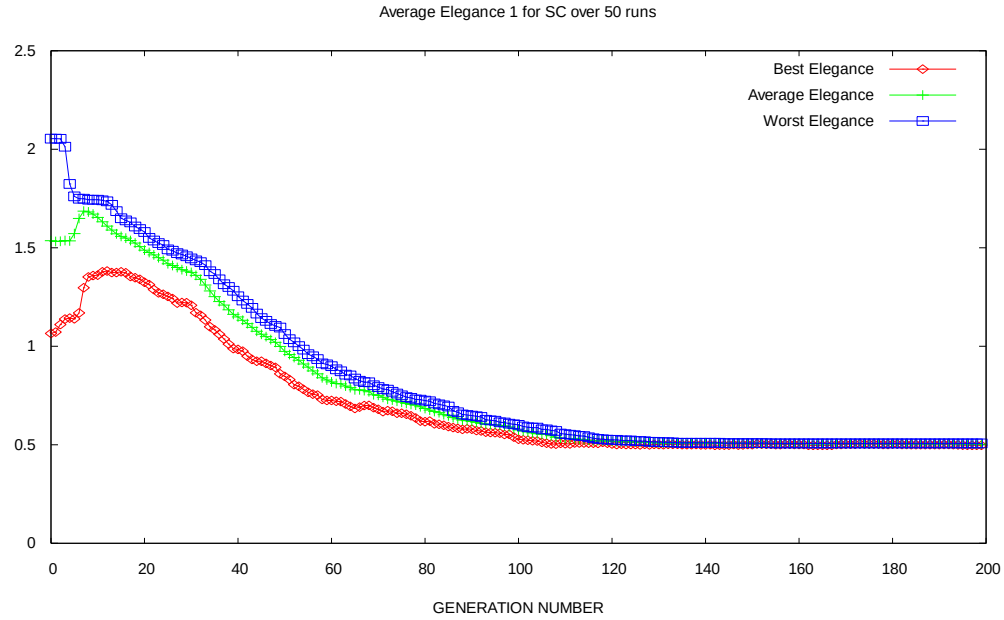

Figure 8.10. Best, Average and Worst Elegance 1 Values for Select Cruises

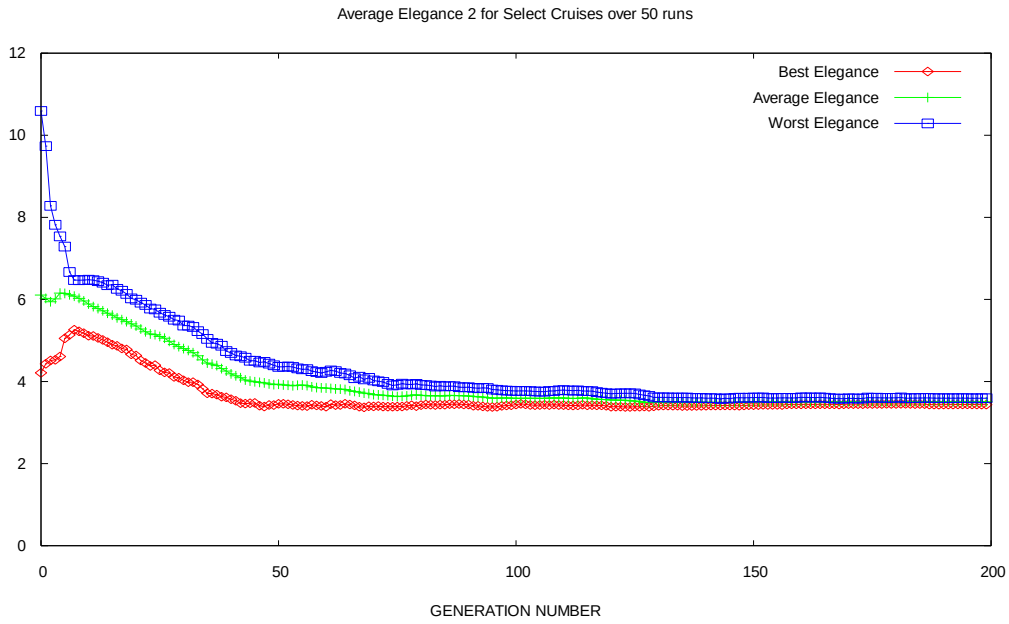

Figure 8.11. Best, Average and Worst Elegance 2 Values for Select Cruises

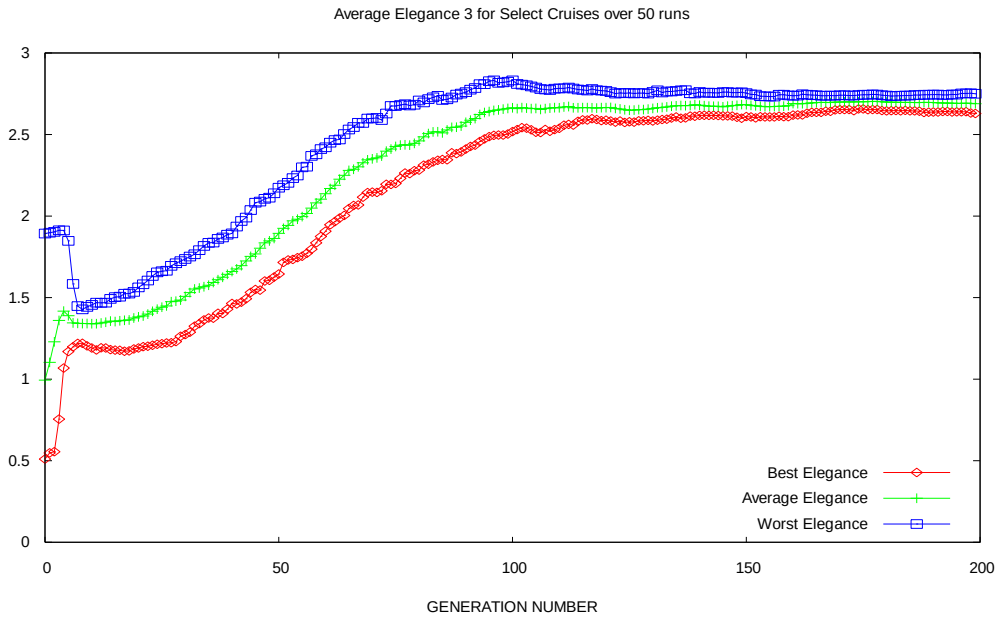

Figure 8.12. Best, Average and Worst Elegance 3 Values for Select Cruises

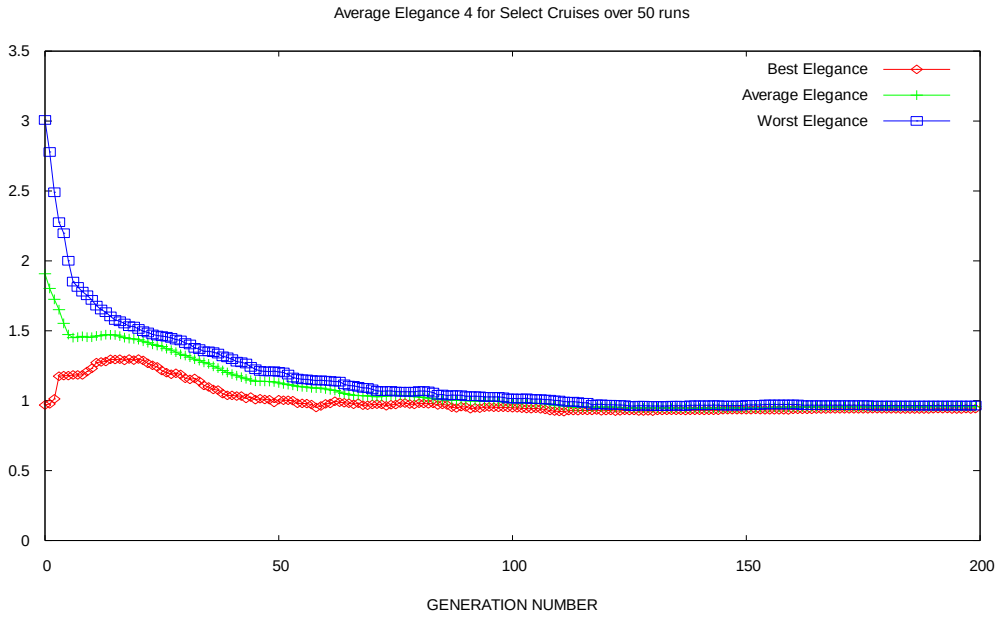

Figure 8.13. Best, Average and Worst Elegance 4 Values for Select Cruises

anything, it is evident that as search proceeds, the results for elegance 3 can worsen appreciably. It is also difficult to discern any effect of design problem scale on software design elegance results – results are broadly similar for all three example software design problems. In one sense, these results are not surprising: if elegance - the symmetry of distribution of attributes and methods among classes - is not used as a basis of selection of superior designs for reproduction into offspring generations during evolution, then evolutionary search could not be expected to arrive at software designs that exhibit superior elegance metrics. Nevertheless, it does reveal that there may or may not be coincidental improvement in elegance values should evolutionary search be steered by quantitative external coupling alone. Thus if elegant software designs are to be arrived at by evolutionary search, this must be the result of human qualitative evaluation of software designs jointly steering the interactive search.

8.4.2 Designer Interaction and the Dynamic Interactive Interval

Across the three example software design problems, the participation of the seven software development professionals with the interactive design framework has been observed for 149 design episodes, involving 1,942 designer interactions in total. Thus the average number of designer interactions per episode is 13.0604 (with a standard deviation of 7.1690). However, different numbers of interactions are observed for each software design problem and these are shown in table 8.3.

Table 8.3. Comparison of Average Number of Interactions for Design Problems

Design Problem	Number of Episodes	Average Number of Interactions / Episode	Standard Deviation
CBS	54	14.9444	7.5521
GDP	54	12.1851	4.1455
SC	41	11.7317	6.2690

It can be seen that as the scale of the design problem increases i.e. from Cinema Booking System (CBS) to Graduate Development Program (GDP) to Select Cruises (SC), the average number of designer interactions decreases. To establish whether differences in the average number of interactions are statistically significant, Student's

t-test has been applied at a 95% confidence level. Results of actual confidence levels in percentages are shown in table 8.4, and it is observed that there is a significant difference between the average number of interactions for the Cinema Booking System and Select Cruises. One possible explanation for this might be that as the scale of the software design problem increases, the cognitive load on the designer increases during design evaluation. Thus as the cognitive load on the design increases, the onset of user fatigue may be brought about earlier.

With regard to the effect of the interval constant on the fitness proportionate dynamic interactive interval, episodes with interval constant values of 1, 5, 10, 15, 20,

Table 8.4. T-test Actual Confidence Levels for Average Number of Interactions

	GDP	SC
CBS	94.62%	**97.02%**
GDP		25.27%

Table 8.5. Average Number of Generations at Designer Halting for Various Values of Interval Constant

Interval Constant	CBS		GDP		SC	
1	19.5714	(8.65750)	12.5000	(9.1378)	13.0000	(7.0000)
5	24.5714	(11.5449)	21.3300	(15.9900)	40.6660	(21.1266)
10	37.1428	(12.3481)	41.3333	(16.7411)	50.0000	(31.0900)
15	38.4285	(9.1443)	55.5714	(25.3367)	79.2000	(37.4860)
20	42.4285	(9.7614)	59.6666	(21.5282)	87.2000	(33.4990)
25	41.5714	(12.1223)	60.7142	(21.0769)	113.6000	(55.5300)
30	42.2500	(8.4850)	72.2857	(22.8108)	110.4000	(57.4400)
35	48.1000	(10.230)	75.5000	(29.6400)	149.8830	(40.2800)
40	48.2500	(19.4100)	96.0000	(43.1810)	141.5000	(38.1300)

25, 30, 35 and 40 have been observed for the Cinema Booking System (CBS), Graduate Development Program (GDP) and Select Cruises (SC) software design problems. The average numbers of evolutionary generations reached at designer halting are shown with standard deviation in table 8.5. The data are also plotted with standard deviation in figure 8.14.

Table 8.5 and figure 8.14 reveal that the number of evolutionary generations reached at design halting broadly increases with the value of the interval constant, possibly reaching a plateau at higher values. This is in line with expectations as low values of interval constant entail low numbers of generations between interactions, leading to limited exploration during evolutionary search before designer halting. Conversely, high values of interval constant bring about higher numbers of generations between interactions, enabling greater exploration during search. The appearance of a plateau in the number of evolutionary generations for each design problem is interesting in that it suggests that a point has been reached at which an increase in the value of the interval constant does not necessarily bring about greater exploration in terms of generations reached before designer halting. In addition, the effect of design problem scale is apparent. Indeed, larger scale design problems plateau at higher numbers of generations, which suggests that greater exploration (in terms of generations evolved) is required for interactive search of large scale software design problems.

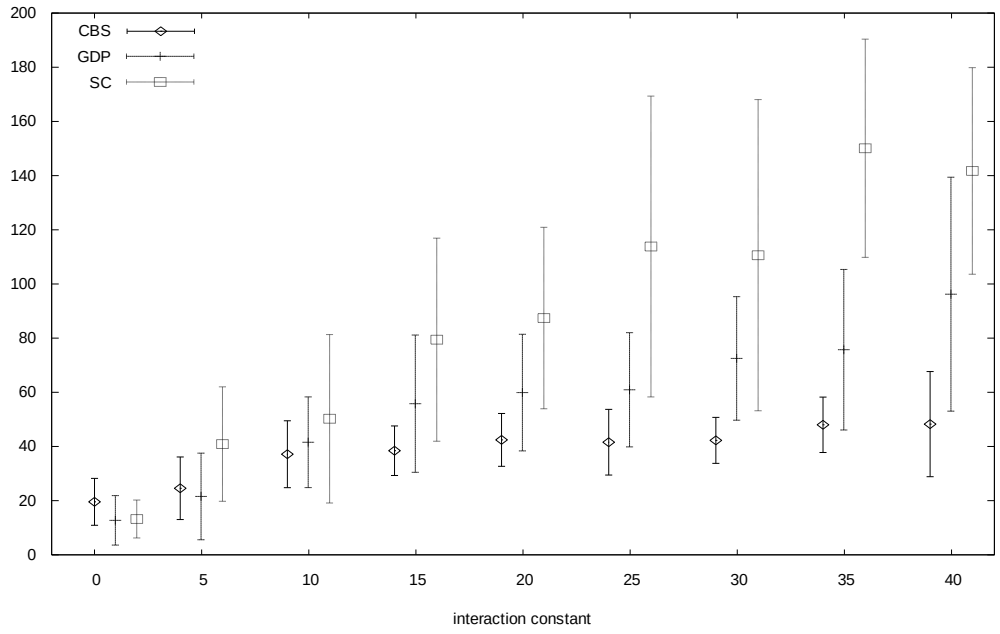

Figure 8.14. Average Number of Evolutionary Generations at Design Halting

Varying values of interval constant might also be expected to influence the number of designer interactions per design episode too. Indeed, if low values of interval constant lead to less exploration during search, it is reasonable to anticipate the early onset of user fatigue as useful design discoveries may be infrequent. On the other hand, if high values of interval constant lead to greater exploration during search, it is also reasonable to expect that useful design discoveries may help prevent user fatigue. To investigate this expectation, the effect of interval constant values of 1, 5, 10, 15, 20, 25, 30, 35 and 40 has been observed for episodes with the three example design problems. The average numbers of interactions per design episode (with standard deviation) are shown in figure 8.15. However, it is unclear from figure 8.15 what effect, if any, differing values of interval constant have upon the average number of interactions per design episode. To determine if any significant differences exist between the numbers of interactions for each software design problem, data have been analysed using an analysis of variance for a one-way, between-subjects design. Results indicate that the mean numbers of interaction do not significantly differ between levels of interaction constant for CBS ($p = 0.407$), GDP ($p = 1.000$) and SC ($p=0.994$). Details of calculations are given in Appendix E.

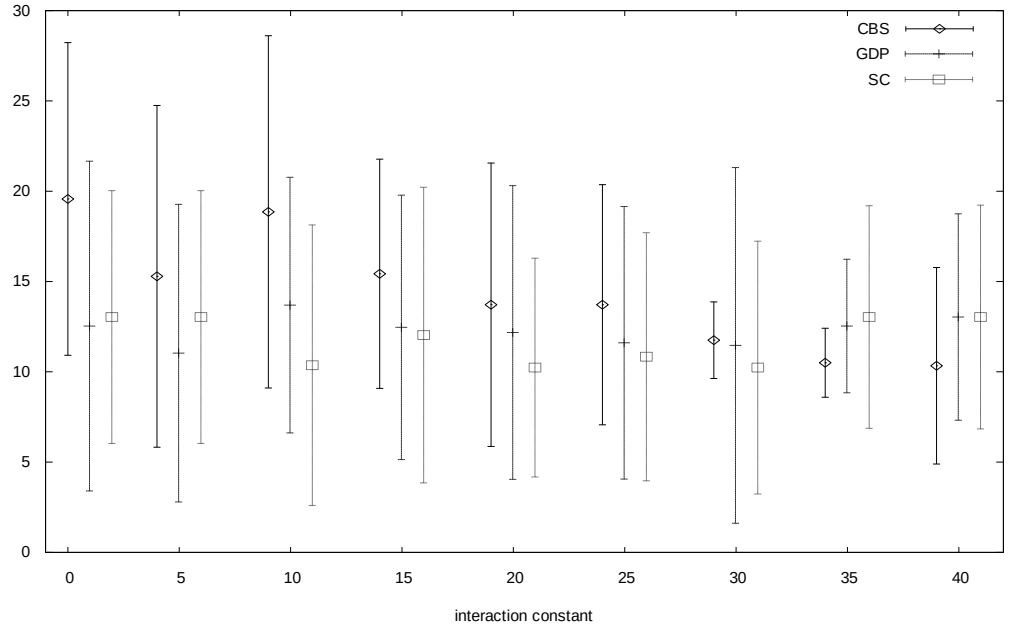

Figure 8.15. Average Number of Interactions for various Interval Constant Values

This finding is both interesting and significant. It is logical to expect that low values of the interval constant might lead to early user fatigue as search does not necessarily arrive at software designs of superior coupling and elegance, while high values of interval constant might result in better exploration and so a more engaging experience of the designer. However, the results seem to suggest that this is not the case. Rather, the results suggest that for this particular interactive software design context, participants have a rather fixed attention span or focus, independent of any design discoveries during the interactive search episode. It is therefore suggested that the duration of the attention span is more influenced by the software design problem and interactive design framework context than the discovery of useful and interesting software designs. This finding does, however, also suggest that for this interactive software design framework at least, the human experience is highly engaging overall, since less successful design episodes endure for similar numbers of interaction as highly successful episodes. Of course, the success of a design episode also depends upon the quantitative and qualitative fitness of the software designs as the design progresses. Results of such fitness values obtained are described in the following two sections which reveal the findings related to design external coupling fitness and designer reward.

8.4.3 External Coupling

The findings of figure 8.14 suggest that for increasing values of interval constant, the longer evolutionary search progresses in terms of number of generations, thus greater exploration is enabled within the interactive search process. It is therefore logical to expect that greater exploration of the design search space might arrive at software designs of superior fitness. To investigate this, average population external coupling values recorded at designer halting have been observed for interval constant values of 1, 5, 10, 15, 20, 25, 30, 35 and 40. The results obtained are shown in table 8.6 and plotted graphically in figure 8.16.

Table 8.6. Average Population Coupling Fitness for various Interval Constant Values

Interval Constant	CBS		GDP		SC	
1	0.5707	(0.1140)	0.6996	(0.0249)	0.8770	(0.0080)
5	0.4718	(0.1474)	0.6175	(0.0613)	0.7290	(0.0960)
10	0.3597	(0.1333)	0.5245	(0.1039)	0.6660	(0.1775)
15	0.3177	(0.1103)	0.478	(0.1183)	0.6322	(0.1580)
20	0.2914	(0.0890)	0.4061	(0.1355)	0.5962	(0.1349)
25	0.2588	(0.1195)	0.4075	(0.1063)	0.5936	(0.1410)
30	0.2020	(0.0162)	0.4057	(0.0842)	0.5640	(0.0700)
35	0.2220	(0.0940)	0.3057	(0.0416)	0.4795	(0.0783)
40	0.2027	(0.0224)	0.3292	(0.0420)	0.4495	(0.0828)

It is apparent from the results in table 8.6 and figure 8.16 that overall, as values of interval constant increase, values of average population external coupling decrease i.e. average population fitness increases with respect to external coupling. In addition, an inverse plateau effect can also be seen for the two smaller scale design problems (Cinema Booking System and Graduate Development Program), although this effect is

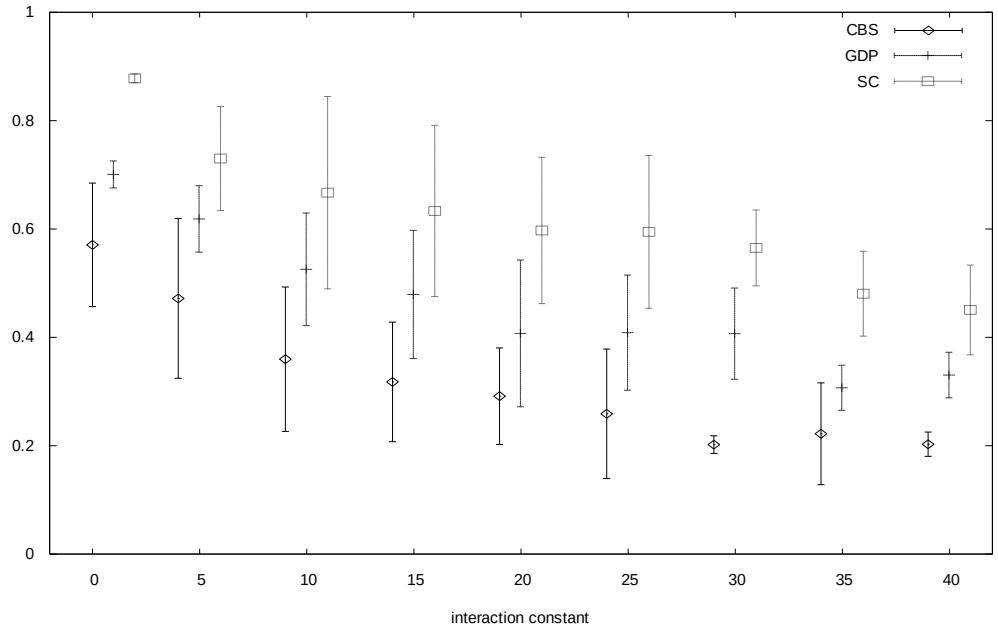

Figure 8.16. Average Population Coupling for various Interval Constant Values

less readily apparent for the large scale design problem (Select Cruises). These findings, when taken with previous findings on evolutionary generation at designer halting, suggest that for quantitative coupling fitness at least, the fitness proportionate interactive interval is an effective component of interactive search for the discovery of useful and interesting software designs. Overall, results obtained so far from the three design problems suggest that a range of interval constant values from 25 to 40 is effective for interactive, evolutionary search of early lifecycle software designs. Small scale design problems might benefit from a lower value in the range, while larger scale design problems might profit from a higher value in the range. Thus a value of 25 - 30 might be suitable for a small scale design problem such as the Cinema Booking System while a value of 35 - 40 might be suitable for larger scale problems such as the Graduate Development Program and Select Cruises. Nevertheless, these findings must be tempered in the light of qualitative designer evaluation and its impact on interactive search. Therefore results of experiments into qualitative designer evaluation and dynamic multi-objective search are described in the following sections.

8.4.4 Reward

Figures 8.17, 8.18 and 8.19 show the average total reward obtained by qualitative evaluation from trial participants at each interaction for the Cinema Booking System, Graduate Development Program and Select Cruises respectively. On each figure, different average total rewards are shown for interval constant values of 1, 5, 10, 15, 20, 25, 30, 35 and 40. It is evident that considerable variation exists at higher numbers of interaction; this is caused by the differing numbers of interactions in an interactive design episode seen among differing participants. It is also evident from figures 8.17, 8.18 and 8.19 that overall, mean reward increases with increasing values of interval constant as participant interactions progress. Such increases in mean reward suggest that increasingly interesting, useful and elegant software designs are being discovered and presented to the trial participants for qualitative evaluation.

For the Cinema Booking System design problem, figure 8.17 reveals that the greatest total reward is obtained for an interval constant value of 20 at least until interaction number 10, although greater reward is obtained for an interval constant value of 35 until generation 12. Thereafter the average total reward is variable due to the varying numbers of interactions observed in the participants' design episode. For example, the marked drop in average total reward for interval constant value 1 from

generations 26 to 27 is due to one individual participant who evaluates designs generously halting an episode. For the Graduate Development Program, figure 8.18 clearly reveals that the greatest total reward is achieved for an interval constant value of

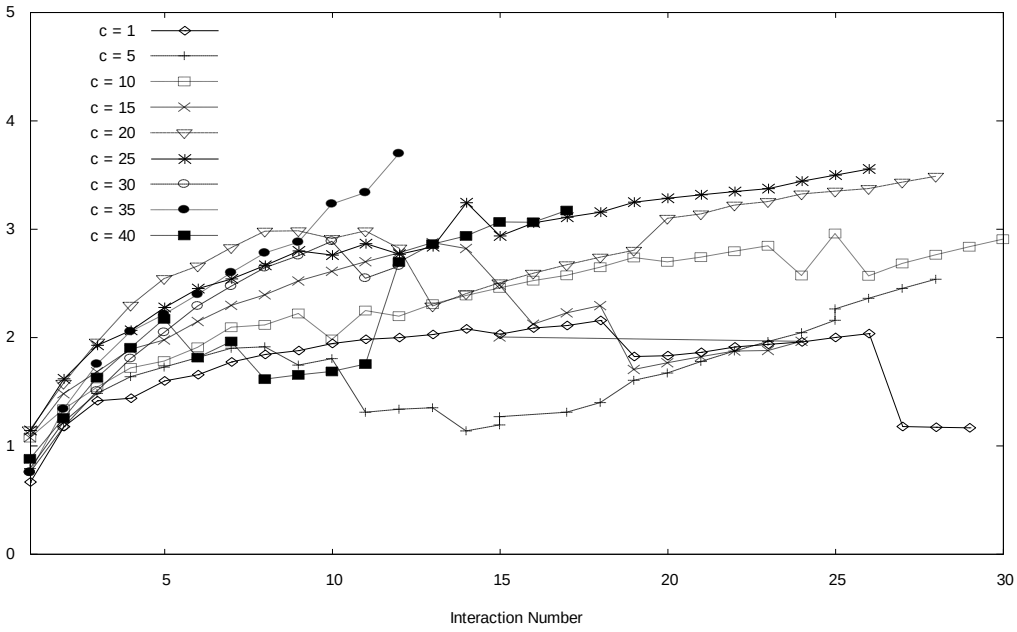

Figure 8.17. Average Total Reward at each Interaction for Various Values of Interval Constant with Cinema Booking System

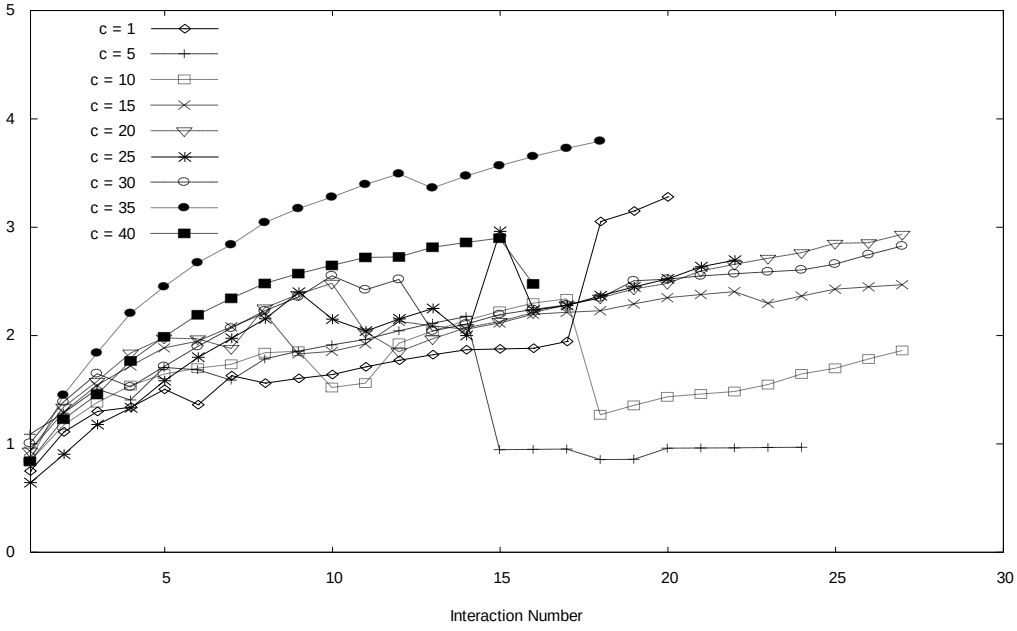

Figure 8.18. Average Total Reward at each Interaction for Various Values of Interval Constant with Graduate Development Program

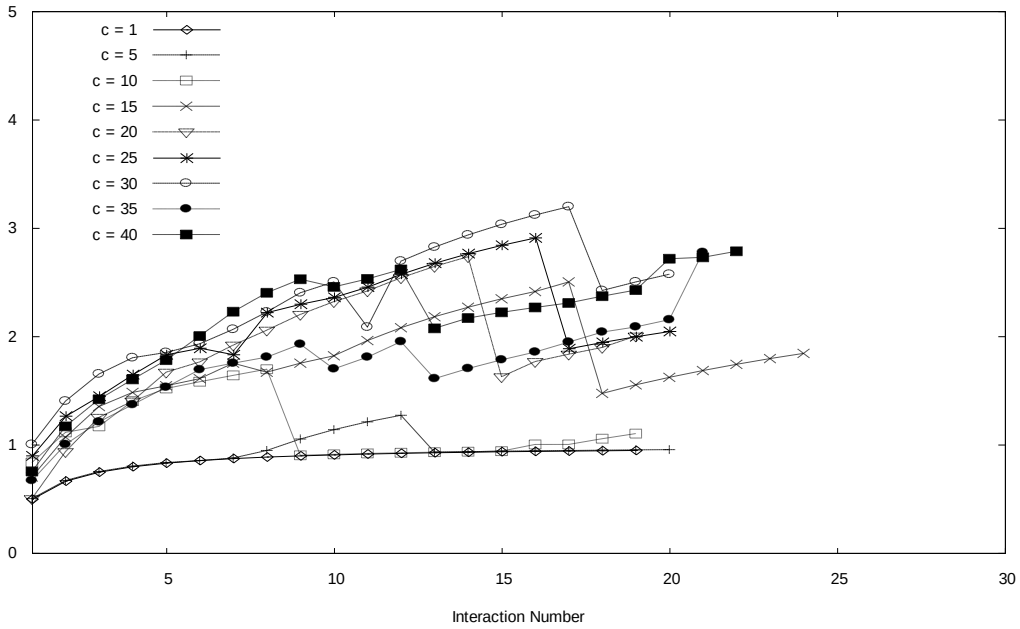

Figure 8.19. Average Total Reward at each Interaction for Various Values of Interval Constant with Select Cruises

35 until interaction number 18. However, results for the Select Cruises design problem show a less clear outcome. Specifically, an interval constant value of 30 produces the greatest participant reward until interaction number 5, then a value of 40 produces greatest reward until interaction 12, only for greatest reward to revert to a value of 30 until interaction number 17. Therefore, results obtained for the Cinema Booking System and Graduate Development Program design problems appear consistent with previous results presented in this chapter. For the Cinema Booking System for example, an interval constant value of 25 produces good exploration (as seen in terms of numbers of generations evolved), good quantitative fitness (as seen in external coupling results), and the greatest average total reward from the trial participants. An interval constant value of 35 produces similarly consistent results for the Graduate Development Program design problem. However, results for the Select Cruises design problem are less clear. Indeed, results from previous sections reveal that the average number of participant interactions per design episode for Select Cruises is lower than for the other two design problems. Taken together, these findings may be due to an increased cognitive load on the participant because of the scale of the Select Cruises design problem leading to semantically complex design visualisations. It seems possible that under increased cognitive load, the participants struggle to assess the elegance of large, semantically rich

designs and so their evaluations of design elegance might become inconsistent over time.

Although the previous three figures reveal the average total reward obtained from participants' qualitative evaluations of software design visualisations, it is also interesting to investigate the contributions of individual elegance metrics in interactive software design episodes. Therefore, the reward obtained for each of the four novel elegance metrics has been recorded, and is presented in tables 8.7, 8.9 and 8.11 for the Cinema Booking System, Graduate Development Program and Select Cruises design problems respectively. In order to see whether statistically significant differences exist between mean reward values for individual elegance measures, t-tests have been performed and the actual confidence levels produced are reported in tables 8.8, 8.10 and 8.12. Where they exist, significant actual confidence levels above 95% are shown in bold.

Tables 8.8 and 8.10 reveal that a statistically significant difference exists between elegance measure 1 and both elegance measures 3 and 4, for both the Cinema Booking System and Graduate Development Program design problems. Specifically,

Table 8.7. Mean Reward for Individual Elegance Measures, Cinema Booking System

Elegance	Interactions	Mean Reward	Standard Deviation
1	226	2.831858	1.292045
2	160	3.018750	1.324524
3	181	3.209945	1.292407
4	156	3.121795	1.235721

Table 8.8. T-test Actual Confidence Levels for Comparisons of Mean Reward for Cinema Booking System

Elegance	2	3	4
1	83.40%	**99.65%**	**97.13%**
2		82.13%	52.45%
3			47.57%

Table 8.9. Mean Reward for Individual Elegance Measures, Graduate Development Program

Elegance	Interactions	Mean Reward	Standard Deviation
1	182	2.56044	1.310466
2	148	2.777027	1.260714
3	131	2.877863	1.258955
4	138	3.021739	1.223059

Table 8.10. T-test Actual Confidence Levels for Comparison of Mean Reward for Graduate Development Program

Elegance	2	3	4
1	57.78%	**96.76%**	**99.85%**
2		49.46%	90.28%
3			65.75%

Table 8.11. Mean Reward for Individual Elegance Measures, Select Cruises

Elegance	Interactions	Mean Reward	Standard Deviation
1	141	2.269504	1.30318
2	103	2.330097	1.183119
3	97	2.536082	1.307525
4	104	2.480769	1.427747

Table 8.12. T-test Actual Confidence Levels for Comparison of Mean Reward for Select Cruises

Elegance	2	3	4
1	29.45%	87.85%	79.04%
2		75.64%	59.05%
3			22.47%

elegance measures 3 and 4 obtained significantly more reward than elegance 1. This is both interesting and significant, as it suggests that some distributions of attributes and methods among classes are qualitatively evaluated as more elegant than others by the participants. Specifically, elegance 3 (the distribution of internal 'uses' among design classes) and elegance 4 (the symmetry of attributes and methods in individual design classes) are evaluated to be more elegant that elegance 1 (the symmetry of attributes and methods among all classes of a design). It is interesting to speculate therefore that the participants valued predominantly class-based symmetry over symmetry of attribute and method distribution for the software design as a whole. This is particularly significant as unlike elegance measures 1, 2 and 4, elegance 3 (the distribution of internal uses in a class) is not indicated in design visualisations and so cannot be visibly discerned by human participants (as noted in section 8.2.1). Because of this, it is speculated that at least for elegance 3, the trial participants' notions of design elegance might perhaps be implicit and intuitive, rather than relating to any visually explicit design constructs. This finding is made more interesting in the light of results of the baseline investigations in section 8.4.1, which seem to indicate that values for elegance 3 become markedly inferior when search is evolved with quantitative design external coupling as the sole objective fitness function without human interaction. In the light of this, it is also speculated that although the participants acknowledge and value the contribution of minimising design coupling during search, they also appear to reward elegance measures that may or may not be in conflict with this quantitative measure.

Figures 8.17, 8.18 and 8.19 also show that use of mean reward for each elegance measure results in prompt increases in reward for approximately the first ten designer interactions of the design episode. In order to investigate the potential impact of this on dynamic selection weightings for the elegance measures, the following section reports quantitative elegance values and dynamic selection weightings during interactive software design episodes.

8.4.5 Elegance and Dynamic Multi-objective Search

During interactive software design episodes, dynamic selection weightings have been recorded and results of typical single individual episodes for the Cinema Booking System, the Graduate Development Program and Select Cruises are shown in figures 8.20, 8.21 and 8.22 respectively. In the light of previous experimental results, an episode with an interval constant value of 25 has been chosen for illustration in figure

8.20, an episode with interval constant value of 35 for figure 8.20, and an episode with interval constant value of 40 for figure 8.22. While the selection weights for each elegance measure are shown as a stacked histogram, the selection weight of external coupling is not shown for the sake of clarity. As shown in equation (8.2) earlier in this chapter, the selection weight for external coupling is 1.0 minus the combined selection weightings of the elegance measures. Therefore the selection weight for external coupling effectively sits atop each histogram bar in figures 8.20, 8.21 and 8.22, effectively making up the remainder of available selection weighting.

The increase in the selection weightings for the four elegance measures shown in figures 8.20, 8.21 and 8.22 has been found to be suitably representative of all design episodes of that interval constant value for the software design problem. In figure 8.20, increases in selection weights for elegance measures occur in step changes at the beginning of the episode, because the fitness proportionate interactive interval brings about infrequent interaction initially. Later, however, increases in selection weightings are more continuous, as designer interaction becomes increasingly frequent in terms of the number of evolutionary generations. Similar performance can be observed in figures 8.21 and 8.22. Values of selection weightings at designer halting are given in table 8.13.

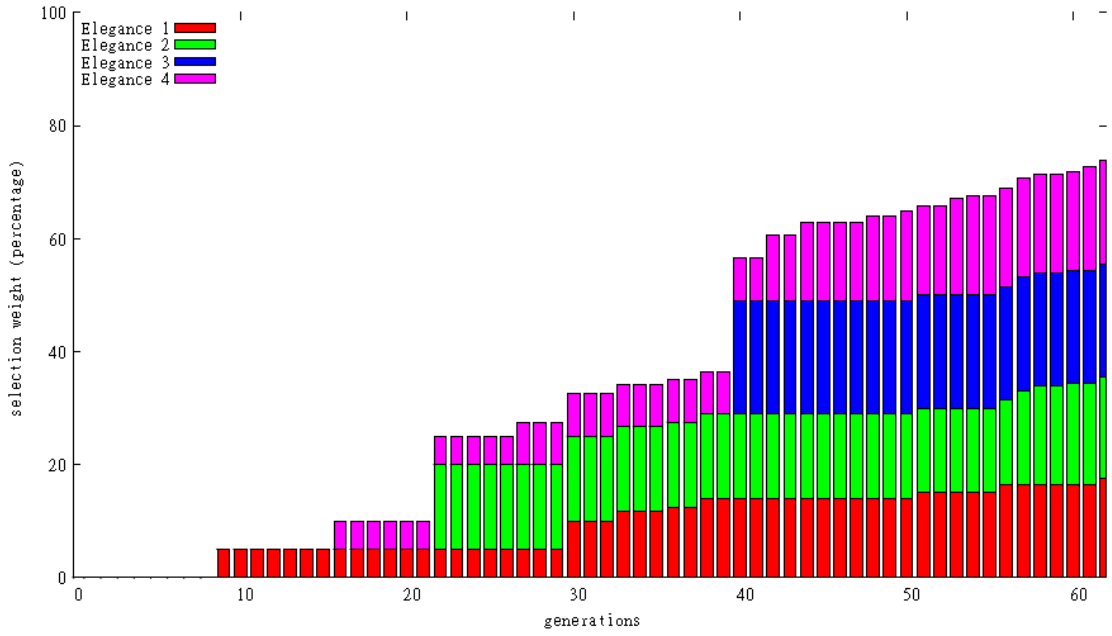

Figure 8.20. Stacked Histogram of Typical Elegance Selection Weights for the Cinema Booking System, interval constant = 25

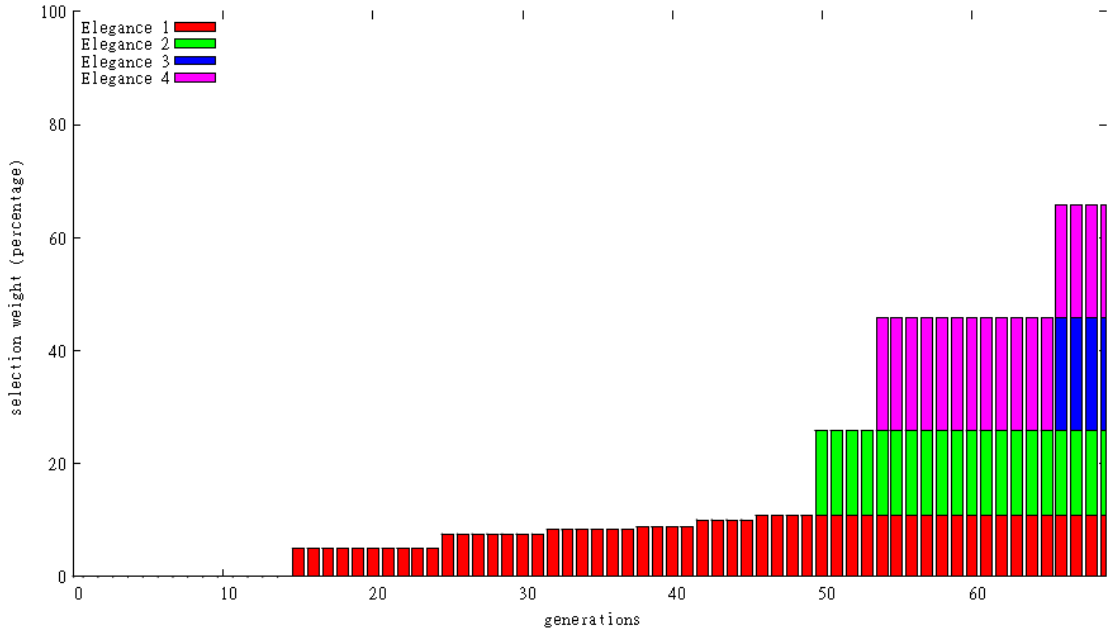

Figure 8.21. Stacked Histogram of Typical Elegance Selection Weights for the Graduate Development Program, interval constant = 35

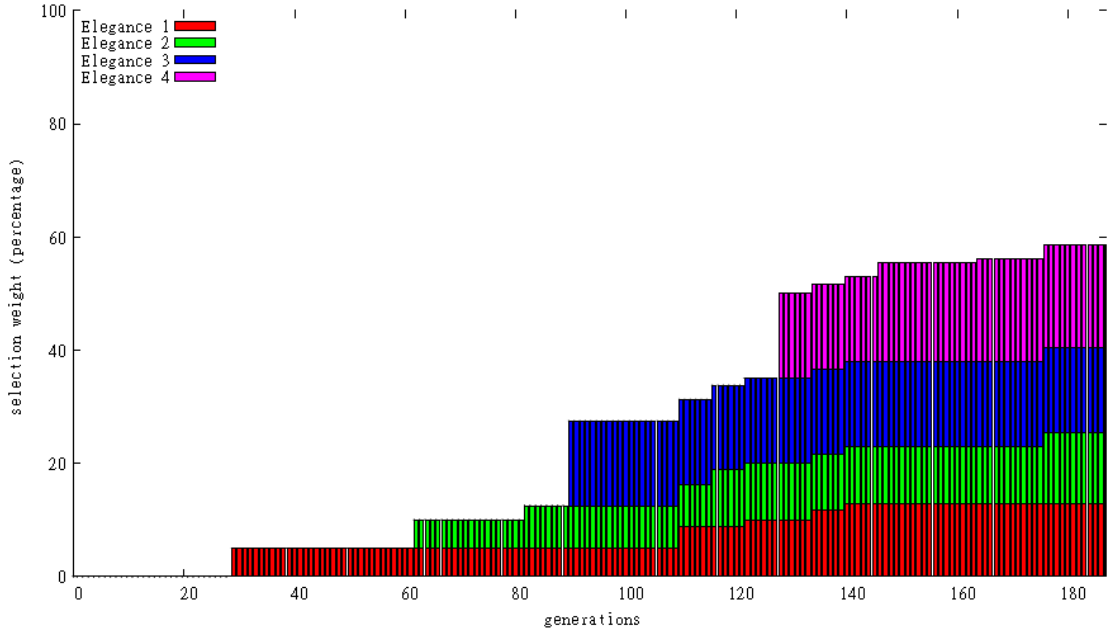

Figure 8.22. Stacked Histogram of Typical Elegance Selection Weights for Select Cruises, interval constant = 40

Table 8.13. Evolutionary Generation Selection Weightings at Design Halting shown in Figures 8.9, 8.10 and 8.11

	Generation	w_{e_1}	w_{e_2}	w_{e_3}	w_{e_4}	w_C
CBS	62	0.175	0.180	0.200	0.183	0.262
GDP	69	0.108	0.150	0.200	0.200	0.342
SC	187	0.129	0.125	0.150	0.183	0.413

It is interesting to note that figure 8.21 reveals mostly unchanging selection weights for each elegance measure from approximately generation 50 to designer halting. This is because the reward obtained from qualitative designer evaluation has been found to be unchanging during these generations too. Therefore, taken as a whole, the results presented in figures 8.20, 8.21 and 8.22 are consistent with previous findings and appear to reveal a picture of dynamic multi-objective interactive localised search wherein the selection weightings of elegance measures respond in a timely manner to the reward obtained from the software designer. Lastly, it is interesting to examine the quantitative elegance values obtained during interactive localised search. Figures 8.23 and 8.24 reveal two example design episodes for the Cinema Booking System in which values

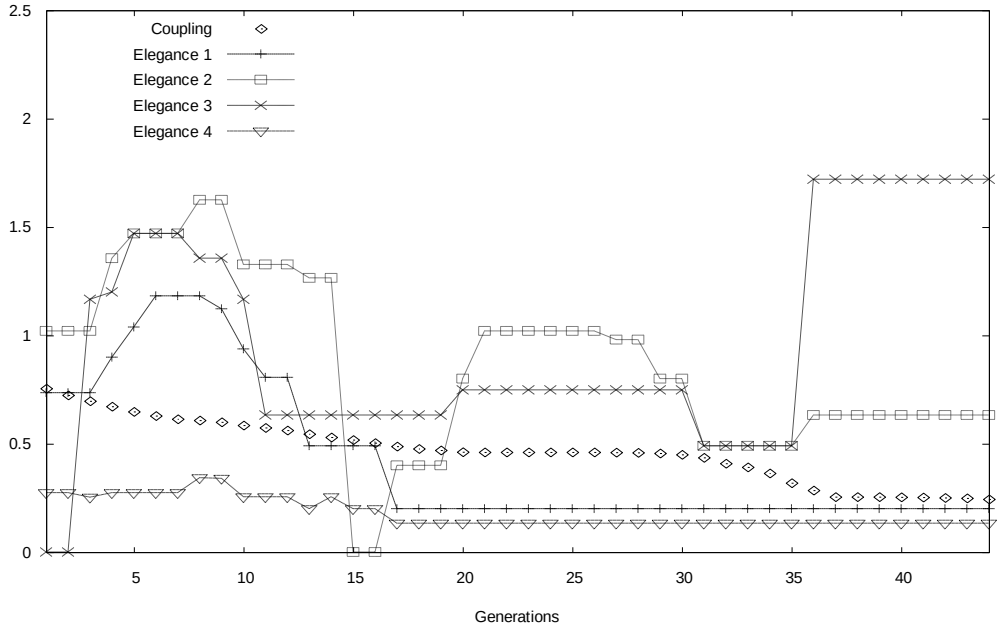

Figure 8.23. Example 1 of Quantitative Elegance Measures during a Design Episode for Cinema Booking System, interval constant = 25

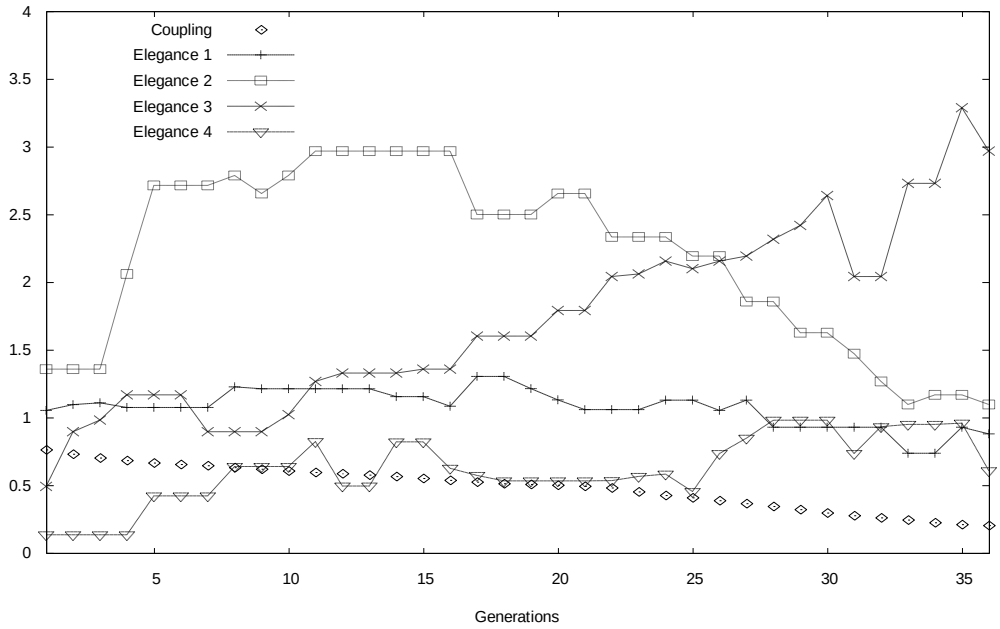

Figure 8.24. Example 2 of Quantitative Elegance Measures during a Design Episode for Cinema Booking System, interval constant = 25

for average population external coupling and the best elegance value in population for each quantitative elegance measure are shown. Based on previous results, an interval constant value of 25 has been again selected for the two example design episodes. No attempt has been made to normalise the quantitative measures – raw data are plotted as recorded. The two examples provided are considered sufficiently representative of the bulk of design episodes drawn from the three example design problems; for the sake of brevity, further examples have not been included.

It is apparent from figures 8.23 and 8.24 that considerable variability exists in the fitness of the quantitative elegance measures as the design episode progresses. In early generations, this is explained by relatively low selection weights for elegance measures. Indeed, if external coupling is the sole or predominant fitness measure used to select offspring, it might reasonably be expected that values for elegance measures would be variable. However, at later generations, as selection weights for elegance measures increase, it might also be expected that fitness values for elegance may improve. Of course, in a multi-objective dynamic search, elegance fitness measures can conflict, as the results of the baseline experiments in section 8.4.1 indicate. Thus as can be seen at later generations in example 2, figure 8.24, although elegance 2 fitness increases, elegance 3 fitness deteriorates. In example 1 in figure 8.23, elegance 3 fitness also deteriorates as design halting is approached. This result is interesting, especially as

this might appear to be at odds with the previous finding that elegance 3 obtains significantly more reward from software designers than elegance 1. However, it is instructive to recall that other authors (Tagaki, 2001, Shakelford, 2007, Caleb-Solly and Smith, 2007, Machwe and Parmee, 2009) have reported the possibility of loss of linearity of focus over the trajectory of time during interaction. Thus it is also possible that designer qualitative evaluation itself might at times be inconsistent, which is perhaps not surprising for such an abstract concept such as design elegance. Furthermore, it is likely that qualitative evaluation depends on the overall human experience of interaction, which includes any pre-existing personal experience and perspective that the designer brings to the situation. Therefore, observations on the human experience of software design discovery with the interactive, evolutionary search-based interaction are reported in the following section.

8.4.6 Human Experience

It was found in the empirical study reported in the previous chapter that the participants were keen to express opinions relating to their experience of interacting with the early lifecycle software design interactive framework, including suggestions for possible enhancements. Building on this, participants in the qualitative elegance investigations have been invited to provide any comment on their human experience of the interactive design episodes. Four of the participants took up the invitation; their comments are provided in Appendix F of this thesis and an analysis provided as follows.

Unlike in the empirical study in the previous chapter, the participants did not comment on any software designs appearing unnatural e.g. designs with 'Blob' classes (see section 8.1). The participants appear to have regarded the software designs visualised for evaluation as natural in appearance. Indeed, overall, the participants found the interactive software design episodes to be very engaging. Participant 3 reports that *"I found the sessions quite enjoyable, the enjoyment gained from working towards, and seeming to achieve, a useful goal: i.e. a good design"*. It is interesting to note that at times, participant 3 felt that they were attempting to 'encourage' the interactive framework to design discovery by providing reward. Participant 7 reports that *"I found the tool to be surprisingly engaging, so much so that, at times, I think I lost sight of the aim of the task and was more focussed on the looking to see how the results changed from one output run to the next"*. This is certainly consistent with the findings of section

8.4.2, in which it is found that the number of designer interactions per episode is independent of the interval constant value (see figure 8.15).

The participants also report the effectiveness of the graphical visualisations of software designs, and highlight the use of colour. Participant 5 remarks that *"the use of colour was a heavy influence"*. Participant 7 comments that *"... colours had a huge impact on my decision making"*. Participant 6 reports that with respect to the graphical interface of the interactive framework, *"the interface of the tool is very easy to follow. The idea of being able to visualise the degree of cohesion and coupling is very good. I believe the tool helps the users to easily understand the quality of the software design."* Nevertheless, despite the effectiveness of software design visualisation, the participants have also provided insights into the at times uncertain relationship between the abstract concept of design elegance and quantitative design cohesion and coupling. For instance, Participant 2 reports that *"I'm not convinced that I really made any 'elegance' judgements – my judgement was principally guided by the tool"*. Participant 5 also comments on feeling uneasy and daunted by judgements of design elegance: *"I felt that sometimes that my judgement values altered during the course of a run – especially for the more complicated examples. So... [sic] the perceived lack of consistency could undermine the confidence in the value of the decisions"*. Such a lack of consistency in elegance judgement may go some way to explain the variability in the quantitative elegance measure results obtained in the previous section, 8.4.5.

Building on these comments, the issue of design problem scale also seems to have had an impact on the participants. Participant 3 comments that for the Select Cruises design problem, *"this was so complex that I needed to write down (i.e. record using pencil and paper) both the coupling score and an assessment of the overall cohesion in terms of the numbers of red classes, the number of yellow, and so on. This was because from one trail to the next, I could not remember precisely enough how good (or bad) the coupling of cohesion had been"*. Further evidence of the impact of design problem scale can also be found in a comment from participant 7: *"I found the cruising problem extremely difficult to concentrate on. It was very difficult to absorb the very rich information presented by the tool"*. It thus seems likely that the high cognitive load of evaluating the large scale Select Cruises design problem impacts both on designer engagement in terms of a reduced number of interactions per episode (see table 8.3), and consistency of participant focus during interactive episodes.

With respect to possible enhancements for the interactive framework, participant 7 comments that *"in general I guess I'd have liked even more interactivity"* including the ability to modify attribute and method allocations to classes via a 'drag 'n' drop' graphical facility. It is speculated that such a capability would be complimentary to the self-adapting mutation of the search process rather than constituting a predominantly manual search process. Participant 7 also comments that they *"couldn't save any 'good corners' of the designs generated"*. This is interesting because although for the purposes of the empirical study in the previous chapter the interactive framework provides the designer with the ability to place interesting and useful designs in a portfolio (or archive), this capability might usefully be extended to include discoveries of partial software designs. Indeed, the search process might thus be able exploit the portfolio archive as part of the dynamic, multi-objective evolutionary search.

8.5 Conclusions

The 133 year's experience of professional software development among the seven participants in the experiments into qualitative elegance evaluation appears to indicate that the participants possess a high level of competency for early lifecycle software design. Such a high level of design competency would suggest a level of credibility for their elegance evaluations. It also suggests that the participants are representative of a meaningful segment of professional software engineers practicing today. Overall, participant comment on their interactions with the interactive evolutionary search and exploration framework is highly positive, with a number of participants stating how much they enjoyed the interactive experience.

Experimental results and participant comment suggest that use of the four novel quantitative elegance measures has been effective and useful in producing natural looking software designs for visualisation within an interactive design episode. This production of natural looking software designs is an important prerequisite of compelling designer engagement in interactive search and exploration. Of course, there are other components necessary for designer engagement, which relate to the effective integration of qualitative designer evaluations of design elegance with quantitative fitness functions within interactive search. Moreover, it is concluded that the effective integration of all these components is the crucial factor in achieving an engaging interactive design experience to enable the discovery of useful and interesting early lifecycle software designs. Thus in addition to the four novel elegance measures, these

further components include a dynamic, fitness proportionate interactive interval, and reward-driven selection weightings within dynamic multi-objective evolutionary search.

Results of experiments into the use of a dynamic, fitness proportionate interactive interval reveal that a judicious balance between the conflicting multiple objectives of quantitative coupling and qualitative design elegance has been achieved. It is concluded that an effective range for values of interval constant lies between 25 and 40. For the design problems investigated in this study, values of 25 to 30 are appropriate for the Cinema Booking System, while values of 35 to 40 are appropriate for the Graduate Development Program and Select Cruises design problems. This suggests that a higher value of interval constant is appropriate for larger scale design problems. For other design problems generally, perhaps where the problem scale is not known, a value of 35 is suggested to provide a robust default interval constant for initial search and exploration. In addition, it is interesting to note that participant software designers exhibit a fixed-length attention span for interactive design episodes with a specific design problem, regardless of the value of interval constant. This finding is somewhat unexpected. It is logical to surmise that low values of interval constant, where designer interaction is frequent, might lead to poor search and exploration, which in turn results in the onset of user fatigue. However, experimental results show that the number of designer interactions per episode is independent of the interval constant. It is concluded therefore that the interactive design context and pre-existing individual designer experience and perspective have great impact. Furthermore, it is also concluded that this is evidence of a compelling experience with the interactive framework – even in interactive design situations where successful design discovery is unlikely, the participants continue to interact for similar length episodes as in the most successful episodes.

It is also concluded that dynamic multi-objective search is an effective component of the engaging interactive software design experience. At appropriate values of interval constant:

- the number of evolutionary generations is judiciously progressed at designer halting,
- average population coupling values reach superior fitness,
- individual elegance star-ratings improve in a timely manner to produce increasing reward, and

- dynamic selection weightings reflect increasing reward in a timely manner.

When taken together, it is concluded that dynamic, multi-objective evolutionary search and exploration is effective at enabling software design discovery designs while reflecting designer elegance intentions. With regard to overall reduction of user fatigue, all these aforementioned components play their part, but it is their integration within an interactive framework that makes them effective.

Software design elegance appears to be an important but complex factor in interactive evolutionary search and exploration. On the one hand, the notion of design elegance is well understood by software designers. Indeed, it is widely held that the formulation of elegant software designs that are robust yet flexible to inevitable change is an essential and crucial part of effective software engineering and software development (Wirfs-Brock, 2007, Boehm and Beck, 2010). Therefore, the power of the quantitative novel elegance metrics has been exploited in the selection of natural looking designs for designer visualisation and dynamic, multi-objective search and exploration. On the other hand, design elegance is an abstraction, existing in designer thought. Sometimes, design elegance cannot be explicitly articulated. In such situations, visualisation of software designs is crucial, and may well influence designer elegance evaluation, the use of colour being a powerful catalyst for abstract designer evaluation. Indeed, it is an interesting and important finding that elegance 3 (the distribution of internal 'uses' among design classes) and elegance 4 (the symmetry of attributes and methods in individual design classes) are evaluated to be more elegant than elegance 1 (the symmetry of attributes and methods among all classes of a design). In short, predominantly class-based symmetry is more highly valued by designers than symmetry of attribute and method distribution for the software design as a whole. It is conjectured that this is an example of the software designers managing the cognitive load of design evaluation by focusing more on the elegance of 'chunks' of the design – individual classes – than the design as a whole. It is also noteworthy that some inconsistency of elegance evaluation has been perceived by some participant designers, especially as design problem scale increases. This is consistent with the view that cognitive load on the designer is also a crucial factor in the onset of user fatigue and linearity of designer focus over time (Tagaki, 2001, Shakelford, 2007, Caleb-Solly and Smith, 2007, Machwe and Parmee, 2009).

Nevertheless, experimental results also indicate that there are some limitations to the effectiveness of the interactive framework and the proposed interactive search

approach. Firstly, the design context is crucial. The discovery of useful, interesting and elegant software designs is only as effective as the interactive experience will allow. Thus the interactive context of the designer, bringing their individual experience and perspective, combined the interactive framework with its design visualisation, imposes a limit on productivity. Secondly, the scale of the example design problems is also a limiting factor. Indeed, the Select Cruises design problem appears to be at the limit of effectiveness for the proposed approach. At such large scale design problems, cognitive load is high and may limit consistency of design evaluation. Thirdly, while the learning mechanism of calculating mean reward for each of the elegance measures is effective and straightforward, other learning mechanisms may promote more sensitivity within the constrained numbers of designer interactions. For example, it may be possible to adopt a rolling or windowed average reward derived from the most recent 3 interactions. Alternatively, it may also be possible to use an incremental implementation of mean reward by providing an estimate of average reward (e.g. Sutton and Barto, 1998).