9 CONCLUSIONS

Early lifecycle software design is a people-intensive, non-trivial and demanding task for software engineers to perform, and a fundamental activity of software development. Many modern ‘agile’ software development methodologies respect the importance of human involvement in software development and have adopted incremental delivery approaches to manually ‘evolve’ the software under development (Boehm and Beck, 2010). Yet the difficulties inherent in early lifecycle software design endure. Furthermore, existing computational support has limitations, not least a lack of support for cognitive activities during software design. However, computational intelligence (such as evolutionary computation and machine learning) has been applied with promising results in support of interactive search and exploration in a range of other design fields. Therefore, this thesis is based on the hypothesis that computationally intelligent tool support can greatly assist the designer with the complexities of discovering design concepts and information relevant to early lifecycle software design.

9.1 Representation

The aim of computational search and exploration in early lifecycle software design is to enable the discovery of useful and interesting software designs expressed using Unified Modelling Language (UML) class models which are demonstrably traceable to the stated software design problem. To underpin such evolutionary search and exploration of the early lifecycle software design solution space, it is necessary to represent the design problem and the design solution such that they are not only sufficiently abstract for human comprehension and but also a natural fit for computational evolutionary search. A novel object-based representation is proposed in chapter 4 and comprises methods and attributes allocated among classes. Indeed, this discrete object-based representation provides a natural and appropriate abstraction of the essential characteristics of the software design solution, and directly facilitates effective visualisation for the interacting human designer. Furthermore, the object-based representation enables efficient genetic operators and provides a high degree of traceability from design problem to design solution. A further benefit of the object-based representation is that it enables estimation of the cardinality of software design search spaces. Such estimates reveal an exponential growth in the scale of software design search spaces, which seems likely to increase the cognitive load on the software

designer and thus may be one of the many causative factors behind the difficulties of early lifecycle software design.

9.2 Exploration and Exploitation of Software Design Search Space

Initial experiments described in chapter 6 investigate the performance of the discrete, object-based software design representation and its associated genetic operators. Results reveal that using manual parameter tuning, the speed of localised search is highly satisfactory and that colourful class design visualisations are human comprehensible. In addition, external coupling is found to be a more useful and tractable quantitative measure of design fitness than class cohesion, and mutation is found to more computationally efficient and explorative than crossover on the design problems tested. Nevertheless, software design problems arise in a wide variety of software development situations and show a range of scale and complexity. Thus in an attempt to address this range of scale and complexity, it is concluded that as a basis of user-centred, interactive evolutionary search, a more flexible, yet robust and scalable approach is required. Therefore, further experiments into localised search using dynamic parameter control have been conducted for three real-life example design problems. Results show that self-adaptation provides a far wider range of mutation probabilities than could have been anticipated through manual parameter tuning. It is therefore concluded that dynamic parameter control via self-adaptation of mutation probabilities provides a more flexible, robust and scalable basis for localised evolutionary search, which in turn might produce useful and innovative early lifecycle software designs for a variety of software design problems.

9.3 Collaborative Designer / Computer Interaction

In any interactive evolutionary computation system, it is vital to enable a natural, collaborative interaction between the human designer and evolutionary search. To enable this, a number of software agents have been formulated to provide an interactive framework. By means of this framework, collaborating software agents and the software designer interact jointly to steer the direction of evolutionary search. As described in chapter 7, the notion of an interactive design episode has been built upon to provide a context for software agent-based interaction. At the beginning of a design episode, a global, multi-objective interactive search is conducted, which allows the designer to identify promising localised ‘zones’. The number of classes in a software design is

important, and so localised zones, in which all software design solutions comprise the same number of classes, are selected for subsequent localised search. It is concluded that within an early lifecycle software design episode, this provides an effective, natural and interactive mechanism to narrow and focus the search to promising localised zones.

At this point in the thesis, an empirical investigation of the interactive framework of software agents conceived thus far is conducted. As described in chapter 7, although the investigation is of small scale, interesting findings have emerged. For example, it is observed that when working with the evolutionary search-based interactive framework, generation of candidate software designs is abundant while evaluation of the candidate designs is greatly assisted by quantitative fitness values presented to the designer. In addition, trade-off analysis of software designs is greatly enhanced. Significantly, the colourful visualisation of UML designs is found to hold designer engagement, and when combined with enhanced generation of multiple candidate designs, enables periods of designer reflection thus enabling opportunities for sudden design discovery. It is concluded that a natural, collaborative interaction between human designer and the interactive framework of search-based software agents has been achieved; such a natural, collaborative interaction appears to be effective in promoting design discovery. It is also noticeable that the participants of the empirical study enthusiastically suggested possible enhancements to the capabilities of the interactive framework, not least the opportunity to provide a qualitative evaluation of the 'quality' or 'appearance' of the software design.

9.4 Elegance and Dynamic Multi-objective Localised Search

Responding to these suggestions, ways have been formulated to enhance the interactivity of the framework with respect to the 'quality' or 'appearance' of software designs. To facilitate this, four novel software design elegance measures are proposed in chapter 8 relating to symmetry of distribution of attributes and methods both among individual classes, and among the design as a whole. Designer interactivity is enhanced by using the novel elegance measures in two ways. Firstly, elegance measures are exploited to present elegant software designs to the designer for qualitative evaluation, which gives the presented designs a more 'natural' look. Secondly, regarding designer qualitative evaluation as 'reward', a reward-driven dynamic multi-objective evolutionary localised search is enabled. In balance with quantitative measures of design coupling, the dynamic search increasingly weights an average reward for each

elegance measure according to their design reward obtained. In this way, search is increasingly steered toward design solutions reflecting designer elegance intentions. Experimental results suggest that both enhancements of designer interactivity help to reduce user fatigue.

Also mindful of user fatigue, a further enhancement to designer interactivity is the introduction of a dynamic, fitness proportionate interactive interval, which varies the number of generations between each designer interaction. During early search, the interactive interval, in terms of evolutionary generations, is high, emphasising quantitative coupling fitness. However, as coupling fitness improves, the interactive interval decreases. Results of experimentation reported in chapter 8 reveal that higher values of fitness proportionate interactive intervals obtain better reward and facilitate better search and exploration. It is therefore concluded that fitness proportionate interactive intervals contribute effectively to reduction of user fatigue. Further results of experimentation also reveal that when combined with the fitness proportionate interactive interval, the dynamic, multi-objective search obtains increasing elegance reward from the designer, and reflects elegance weightings in a timely manner. It is also significant that designer comment concerning the human experience of interactive search and exploration is highly positive, reflecting good design discovery and even genuine enjoyment at times. Thus with regard to overall reduction in user fatigue, it is concluded that while all the above components play their part, it is their successful combination and integration as an interactive framework that enables the compelling designer experience.

Software design elegance is an important but complex factor within interactive search and exploration. Although elegance plays a crucial part in preventing user fatigue, design elegance is an abstraction, existing in designer thought. It is interesting to note that elegance measure 3 (the distribution of internal ‘uses’ among design classes) and elegance measure 4 (the symmetry of attributes and methods in individual design classes) appear to attract more reward from designers than elegance 1 (the symmetry of attributes and methods among all classes of a design), even though elegance measure 3 has no concrete manifestation in class design visualisations. It is concluded that class-based symmetry is more highly prized by designers than symmetry of attributes and methods among the design as a whole. It is conjectured that that this may be because as the cognitive load of design evaluation reaches a point at which cognitive load exceeds cognitive capability, the designer focuses attention on ‘chunks’

of the design – classes – more than the design as a whole. This is consistent with the mental 'clustering' strategy used by designers as reported, for example, by Lawson (2006) (see chapter 2, section 2.1.1). It is also conjectured that this mental strategy becomes more prevalent with design scale. The larger the scale of the design problem, the more necessary the tactic of 'divide and conquer' becomes in order to manage cognitive load.

9.5 Overall

Taking the results of experimentation in the round, it is concluded that interactive evolutionary computation does indeed afford significant opportunities for effective quantitative and qualitative multi-objective search and exploration of the early lifecycle software design solution space. It is also concluded that the effective integration of software agents and machine learning within an interactive framework is the single most crucial factor contributing to its success. This effective integration has resulted in an engaging and compelling interactive context for the designer to conduct early lifecycle software design. Finally, it is concluded that such computationally intelligent tool support for the software designer enables opportunities for the discovery of useful and elegant UML class designs. It seems likely that the exploitation of such opportunities might lead to better software design traceability, structural integrity and elegance, which may in turn, yield significant software development productivity gains.

Nevertheless, results of experimentation have also revealed some limitations for the computationally intelligently tool support. Firstly, it is evident that the scale of the Select Cruises design problem is at the limit of effectiveness for the interactive framework. In one sense, this might not be totally unexpected as Select Cruises is the largest of the three real-world design problems, with sixteen classes and a search space cardinality of no less than 1.80947×10^{54} (see chapter 4, section 4.4.3). Even with colourful design visualisation, the cognitive load of design evaluation of Select Cruises design solutions is clearly considerable, which might in some part have contributed to non-linearity of designer focus over design episodes. Secondly, although it is clear that an engaging and compelling interactive design context has been produced, all participants in elegance investigations offered suggestions for even greater interactivity. Recurring participant suggestions include the ability to manually 'drag 'n' drop' individual attributes and methods from one class to another within a design visualisation, and the facility to place superior fragments of a design in a portfolio or

archive. It is intended to address these interactivity enhancements in future work. Indeed, the ability to isolate superior fragments (or ‘chunks’) of software designs, and possibly co-evolve elite subpopulations based on the contents of the portfolio / archive seems a useful progression for interactive computationally intelligent tool support. Moreover, this may also enhance natural design interaction, which may in turn help to increase the scale of design problems that might be usefully addressed. Future work will also investigate ways in which reward-based learning might be enhanced to more sensitively reflect designer elegance intentions, and so further reduce user fatigue. Lastly, some elegance investigation participants pointed out that the computationally intelligent tool support also offers considerable potential as an adaptive learning mechanism for software design novices and those coming to early lifecycle software design for the first time. Although this purpose was not conceived at the time of investigation, future work will also investigate this intriguing possibility.