7 EXPERIMENT: EMPIRICAL INVESTIGATION

7.1 Background

For the purposes of empirical investigation, the notion of a design episode is useful for framing a session of early lifecycle software design using an interactive computational support framework. In detailing the components of design thought, Lawson (2004) describes both design events and design episodes. According to Lawson, design events may be the *"physical actions, drawing, modelling, gesturing, acting"* or *"verbalisations … or entirely internal mental operations"*. He goes on to provide examples of various types of design events: *"… a structuring of a problem, a proposition about a possible solution characteristic, a representation of a solution characteristic, an evaluation of a solution characteristic"*. Lawson suggests that design events happen at a point in time and thus are atomistic, with an indivisible nature with respect to design. Lawson further proposes that as events are usually not unconnected, they often exist as part of some larger purpose. A group of events is carried out to move the design forward in some way. Lawson refers to a group of events as an episode thus: *"In a dramatic sense, they consist of a series of transactions that deal with a particular theme or themes that can be used to punctuate a larger narrative into the 'scenes' or 'acts' in plays or operas, or the 'chapters' in books, or the 'episodes' in longer running serials on television or radio. It is not the case that they are entirely discrete and separate from the rest of the narrative but that they seem reasonably self-contained"*. This notion of a design episode appears useful and consistent with observational studies of Guindon (1990) and Curtis *et al.* (1998) and thus is used in this investigation to frame a session of interactive evolutionary upstream software design. In outline, an early lifecycle software design episode might run from start to finish as follows.

To begin, a design episode requires a design problem. Representing design problems as use cases is widely applied in software engineering, and so the narrative text of the use case is used to identify the actions that the software is to perform, together with the data that the software will manipulate. From this, an initial population of candidate classes is derived at random. The designer having provided their search preferences (e.g. population size, maximum number of generations) and chosen from the desired objective fitness functions available (e.g. design coupling, cohesion of classes in a design, number of classes in a design), a software agent performs global multi-objective evolutionary search of the design solution space. To efficiently guide

the designer through the design space, further software agents isolate discrete zones mapping to designs with specific numbers of classes for subsequent local search. At any point in the episode, the designer may inspect an individual design and add it to the portfolio should it appear useful or interesting. The collaborations of the software agents of the interactive framework are described in more detail below. Typically, the first event is to commence logging of design events.

1) Logging of design events is facilitated by an *Event Logger Agent*, which has knowledge of all possible design events, together with the date and time within the episode at which the design event occurs. The Event Logger Agent provides the designer with a design event notification service and a chronological trace of all design events as they occur during the episode, including any preferences and choices of the designer to steer the search.
2) Building on the findings of earlier work of Cvetkovic and Parmee (2002) into agent-based support for design, software designer preferences are held by a *Preferences Agent*. With the design problem recorded, the designer might provide their search preferences for both global and local search. Preferences such as population size, number of generations, choice of fitness functions (and where applicable, mutation and crossover rates) can be supplied by the designer. Alternatively, the designer may opt to select pre-set default values available from the Preferences Agent. The design search space is initialized with a population of candidate class designs. The initial population of candidate class design solutions is created by firstly creating each individual class design with a random number of classes.
3) Multi-objective evolutionary search is enabled by a *Global Search Agent*, which explores the global design solution space, trading-off conflicting objective fitness functions selected by the user. The progress of global search is monitored and controlled by the *Zone Isolator Agent*, which monitors the utility of the evolving population with respect to the trade-off between increasing population fitness and decreasing population diversity. The Global Search Agent terminates global search either at a generation pre-determined by the Preferences Agent, or at a point when the Zone Isolator Agent indicates that halting is judicious. At the termination of global search, each zone is available for local search.
4) After the local zones have been isolated from the multi-objective search, the software designer manually selects promising zones for local search. A list of all

zones is provided to the designer, together with an indication of the diversity present in each zone in terms of the number of unique designs in the zone. The designer may manually select one or more zones to go forward to local search. A *Local Search Agent* initialises the design population of each local search. At the point when global search is halted, it is possible that zones may comprise fewer individual designs than the population size local search preference known by the Preferences Agent. Should this be the case, individual designs in a zone are first placed in the local search population, and then the individual designs are repeatedly cloned and mutated until the local population size preference is reached. This mechanism promotes diversity in the local search. The Local Search Agent then conducts local search using design coupling as the preferred fitness function. Within each local search, the number of classes of each design remains unaltered. Local search proceeds until the number of generations known by the Preferences Agent is reached. To achieve speedy execution time, computational concurrency is exploited by executing each local search agent in parallel. This enables up to approximately 20 local searches to be conducted concurrently. After local search, software design visualisations may be inspected by the designer, who may add useful and interesting designs to the Design Portfolio as required.

The behaviours of all software agents employed in this approach (episode logger agent, preferences agent, global search agent, zone isolator agent and local search agent) are co-ordinated and controlled by an Agent Controller (itself a software agent) that regulates both the life cycles and the task-based behaviours of the agents.

7.2 Software Design Problem Domain

The software design problem domain chosen for investigation is the ‘Graduate Development Program’, which is described in detail in chapter 5, Methodology. The Graduate Development Program is a development of the existing student administration system performed by the in-house Information Systems Department at the University of the West of England, UK. The Graduate Development Program software system seeks to record and manage outcomes relating to personal student development during their studies.

In line with usual practice for the in-house Information Systems Department, initial requirements capture activities have involved regular, highly iterative, people

intensive, interactive sessions with stakeholders where ‘mock-up’ scenarios of usage have been piloted under conference room conditions. While no specific development methodology has been employed, principles common to agile methods (e.g. Beck, 2000) predominate. During the interactive pilot sessions, no computational tool support has been deployed except for rapid construction of mock graphical user interfaces (GUIs). The pilot sessions successfully identified system actors and four main goals that the actors would wish to achieve within a scenario of interaction with the system. The four goals included:

- the ability to record a personal development outcome for an individual student;
- the ability to record personal development outcomes for a batch of many students;
- the ability to generate various reports on personal development outcomes; and
- the ability to export report results in a format capable of being read in desktop spreadsheet applications.

The four goals have been recorded as use cases and are available at Simons (2010b).

7.3 Method and Empirical Investigation Design

The method employed in the empirical investigation is to observe two early lifecycle software design episodes in an industrial setting. The first episode is conducted without the support of the computational framework (i.e. design is performed manually); in the second episode the designers are supported by the interactive framework. The same participants take part in both episodes. Effectively, the manual episode constitutes a baseline (or control) against which comparisons and contrasts with the computational interactive framework supported episode may then be drawn.

The participants being observed include a project manager and business analyst who work within the in-house Information Systems Department under investigation. The project manager and business analyst have been selected for observation as they typically perform early lifecycle software design within the in-house Information Systems Department. The project manager has a bachelor’s degree in Systems Analysis and 20 years professional experience of requirements capture, analysis, design and project management of information systems. The business analyst has bachelor’s degree in Business Information Systems and 7 years professional experience of requirement capture, analysis and design of information systems. Visual and audio recordings of

observations are made for both design episodes, and a textual transcript of verbal utterances has been taken from the recordings. The author is present at both design episodes in order to produce the recordings but remains silent and non-participatory throughout, except with respect to the necessary physical mechanics of producing recordings and tool support in the second episode.

Two issues arise at this point:

- How generalizeable might be the results when the number of participants is small?
- How representative is this sample of the larger population of software engineers?

Given the relative lack of empirical studies reported in the literature for search-based engineering, it is hard to answer these questions. There exists little or no population data to compare this sample against, and there is no standard type of individual who performs early lifecycle software design – education, professional experience, job context and competencies may differ markedly. However, the two individuals selected are held in high esteem by their colleagues, and are representative of some segment of the population of software engineers who perform early lifecycle software design.

The method of the investigation compares and contrasts two design episodes, based on the same problem domain. Clearly, a higher degree of confidence in observations would have been achieved from observing a greater number of participants over further design episodes over different design domains. However, finding suitable people-intensive industrial design situations appropriate to observational studies is not a trivial task. In addition, it is very difficult to ensure that different design domains are comparable in terms of complexity and difficulty of the design problem. Therefore, with respect to method, it is pragmatic to use one problem domain and conduct a manual episode firstly, followed by a tool supported episode. In this way, data obtained from the first episode may therefore be treated as a baseline for comparison with the second. A limitation of this method, however, is that the participants' use of the interactive framework support tool may be influenced by the fact that they have previously engaged with the design problem. The impact of this limitation on analysis of the observational data is discussed later in this chapter in section 7.7, 'Threats to Validity'.

7.4 Observation and Data Collection

Measurements have been selected to investigate the richness of the design episodes both in terms of outputs produced and the means by which the outputs are produced. With respect to the means by which the outputs are produced, a number of characteristics of interaction have been investigated including approaches to:

- concept generation,
- iteration,
- opportunistic realization, and
- medium of interaction.

According to Liu and Bligh (2003), *"...design should contain two types of steps: divergent in which alternative concepts are generated, and convergent in which these are evaluated and selected".* This iterative notion of Liu and Bligh is consistent with reports within software engineering by Jacobson *et al.* (1999) and Glass (2003), who describe software design as a complex, iterative process. Thus divergent and convergent design activities are observed and recorded as a measure of the richness of the design episode. As iteration is widely regarded as a necessary and natural component of design, iteration between not only the problem and solution spaces but also convergent and divergent design activities are observed. Furthermore, sudden discovery moments and opportunistic understandings (as observed by Guindon, 1990) have been noted as being significant events within design episodes and so these are observed too. Finally, as an indicator of the richness of the design episode, the medium of interaction between the two designers has been observed, be it verbal, paper-based sketching, interacting via the search-based support tool, or via UML class modeling.

Textual transcripts of the two episodes are also analyzed according to design mode, design activity, and the occurrence of design events. Design modes and design activities are analyzed within 20 second intervals in the design episode. 20 second intervals have been chosen to provide a reasonable level of granularity of analysis. Design modes include:

- Space – is the design episode focused primarily on the problem or solution space in each 20 second timed interval?
- Thrust – is the thrust of the design episode primarily convergent or divergent in each 20 second time interval?

- Medium – is the medium of designer interaction verbal, sketching, search-based tool supported, or UML class modeling in each 20 second time interval?

Design Activities include:

- Evaluation – are the designers primarily evaluating individual candidate designs in a 20 second time interval?
- Generation – are the designers primarily generating candidate designs in a 20 second time interval?
- Trading-off - are the designers primarily trading-off between multiple candidate designs in a 20 second time interval?
- Scoping – are the designers primarily considering if a candidate design is in scope during a 20 second time interval?
- Reflective silence pauses – have the designers paused for silent reflection?

Design Events are discrete happenings at a point in time in the design episode and include:

- Request for clarification – a designer requests a clarification of design activities of the other,
- Explanation of understanding – a designer explains their understanding of a design activity to the other,
- Sudden discovery – a designer expresses an "ah-ha!" moment of sudden discovery of a design concept or design concept relationship,
- Realization of constraint – a designer expresses a moment of realization that a candidate solution is constrained in some manner by the problem domain requirements,
- Realization of inferred requirement – a designer expresses an insight of an inferred requirement i.e. although not explicitly stated in the case study problem domain specification, a further requirement is inferred as consistent with the specification,
- Inspection of a candidate UML class diagram – a designer inspects a candidate UML class diagram, and

- Add a UML class diagram to portfolio – a designer adds a useful and interesting UML class diagram to the episode portfolio.

7.5 Results

A textual transcript of the baseline manual design episode is available in Appendix A; a transcript of the test design episode is available in Appendix B. In both transcripts, the abbreviation "PM" refers to the participant undertaking a 'project manager' role in the episode; the abbreviation "BA" refers to the participant responsible for a 'business analyst' role.

7.5.1 Duration

It is observed that the duration of the baseline manual conceptual design episode is 37 minutes and 2 seconds (2122 seconds), while the duration of the test design episode with search-based tool support is 55 minutes and 23 seconds (3323 seconds).

7.5.2 Software Designs Produced

It is observed that no design artefacts of early lifecycle software designs are produced during the baseline manual conceptual design episode. While much verbal interaction centered on the explanation of the concept of "Student" and its associated information, no drawings or UML diagrams are arrived at. However, it is also observed that many software designs are produced in the course of the search-based tool supported design episode. Analysis of the transcript reveals that 30 candidate class diagrams are inspected, and from these, 7 are added to the portfolio via the interactive framework support tool. During the test episode, the two participants are observed recognizing a "Student" class after 5 minutes, an "Award" class after 6 minutes, a "Report" class after 16 minutes, and a "Rule" and a "Development" class after 23 minutes. Thus in total, 5 classes are identified in the interactive framework supported design episode, which contrasts with one class identified in the manual design episode.

7.5.3 Event Log

The following is a listing of the Event Log produced by the Event Logging Agent during the test software design episode:

30 July 2008 15:02:49 BST New design episode started. Episode name is: 'lee nick 30 07 2008'

30 July 2008 15:02:59 BST Design problem selected. Graduate Development Programme (GDP)
30 July 2008 15:03:11 BST Multi-objective search parameter selected. MOGA population size set to 100
30 July 2008 15:03:11 BST Multi-objective search parameter selected. MOGA number of generations set to 500
30 July 2008 15:03:11 BST Multi-objective search parameter selected. MOGA search objective function set to external coupling
30 July 2008 15:03:11 BST Multi-objective search parameter selected. MOGA search objective function set to number of classes
30 July 2008 15:03:11 BST Multi-objective search parameter selected. MOGA search priority set to variety
30 July 2008 15:03:17 BST Local search parameter selected. Population size set to 100
30 July 2008 15:03:17 BST Local search parameter selected. Number of generations set to 100
30 July 2008 15:03:17 BST Local search parameter selected. Crossover rate set to 70%
30 July 2008 15:03:17 BST Local search parameter selected. Mutation rate set to 3%
30 July 2008 15:03:17 BST Local search parameter selected. Reproduction / selection operator set to tournament
30 July 2008 15:03:22 BST Isolation of discrete zones selected.
30 July 2008 15:03:22 BST Isolation factor specified. preference of two consecutive falls in utility selected for global search halting
30 July 2008 15:03:40 BST Multi-objective search parameter selected. MOGA population size set to 100
30 July 2008 15:03:40 BST Multi-objective search parameter selected. MOGA number of generations set to 500
30 July 2008 15:03:40 BST Search strategy selected. Search strategy set to multi-objective
30 July 2008 15:03:40 BST Search population initialised. Multi-objective search
30 July 2008 15:03:42 BST Multi-objective search parameter selected. MOGA search priority set to variety
30 July 2008 15:03:42 BST Multi-objective search parameter selected. MOGA search objective function set to number of classes
30 July 2008 15:03:42 BST Multi-objective search parameter selected. MOGA search objective function set to external coupling
30 July 2008 15:03:42 BST Isolation agent specified. two falls
30 July 2008 15:03:42 BST MOGA Search started.
30 July 2008 15:03:46 BST Isolation agent stopped MOGA search. average population coupling is 0.697, sample generation is 57, sparsity count is 0, duplicate count is 3, utility is 0.462
30 July 2008 15:04:45 BST Local search agent starting local search. Agent is not selected to choose discrete zones. Designer has manually selected the following discrete zones: 4, 5, 6, 7,
30 July 2008 15:04:46 BST Local search initatiated. Discrete zone 4
30 July 2008 15:04:46 BST Local search initatiated. Discrete zone 5
30 July 2008 15:04:46 BST Local search initatiated. Discrete zone 6
30 July 2008 15:04:46 BST Local search initatiated. Discrete zone 7
30 July 2008 15:04:56 BST Local search completed. Discrete zone 7
30 July 2008 15:04:57 BST Local search completed. Discrete zone 6
30 July 2008 15:04:58 BST Local search completed. Discrete zone 5
30 July 2008 15:04:59 BST Local search completed. Discrete zone 4
30 July 2008 15:08:09 BST Design added to Portfolio. design 'award class' added.star rating is: 2
30 July 2008 15:10:46 BST Design added to Portfolio. design 'possible student class with high cohesion' added.star rating is: 2
30 July 2008 15:16:12 BST Design added to Portfolio. design 'zone 7 too dispersed' added.star rating is: 1
30 July 2008 15:20:16 BST Design added to Portfolio. design 'z6 d1 number of possible classes' added.star rating is: 3
30 July 2008 15:22:39 BST Design added to Portfolio. design 'z6 d2 possible mix of classes' added.star rating is: 3
30 July 2008 15:27:01 BST Design added to Portfolio. design 'z6 d3 very similar to z6 d1, d2' added.star rating is: 3
30 July 2008 15:33:52 BST Design added to Portfolio. design 'z5 d5 beginnings of student, award, rule' added.star rating is: 3
30 July 2008 15:35:34 BST Local search agent starting local search. Agent is not selected to choose discrete zones. Designer has manually selected the following discrete zones: 4, 5, 6, 7, 5, 6,
30 July 2008 15:35:34 BST Local search initatiated. Discrete zone 4
30 July 2008 15:35:34 BST Local search initatiated. Discrete zone 5
30 July 2008 15:35:34 BST Local search initatiated. Discrete zone 6
30 July 2008 15:35:34 BST Local search initatiated. Discrete zone 7
30 July 2008 15:35:43 BST Local search completed. Discrete zone 7
30 July 2008 15:35:45 BST Local search completed. Discrete zone 6
30 July 2008 15:35:46 BST Local search completed. Discrete zone 5
30 July 2008 15:35:47 BST Local search completed. Discrete zone 4
30 July 2008 15:41:32 BST Design added to Portfolio. design 'z5 d3 student clsss' added.star rating is: 2

7.5.4 Comparison of Design Episodes

Figures 7.1 to 7.6 graphically reveal the results of design modes, activities and events for both observed software design episodes. Following these figures, all observational data is summarised in Table 7.1.

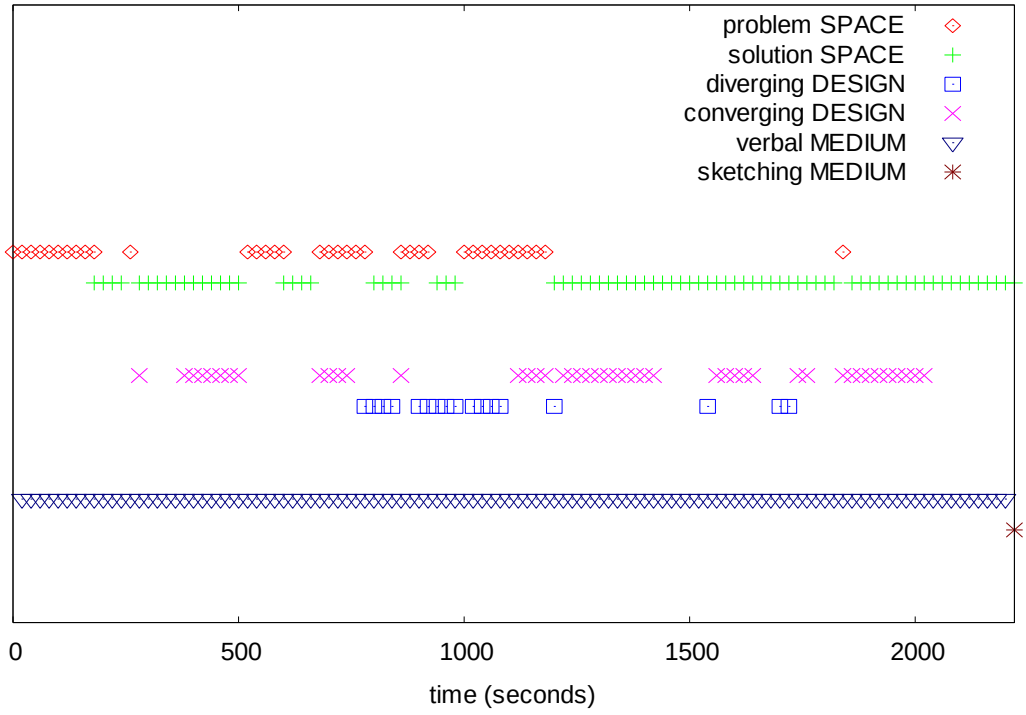

Figure 7.1. Design Modes for Baseline Episode

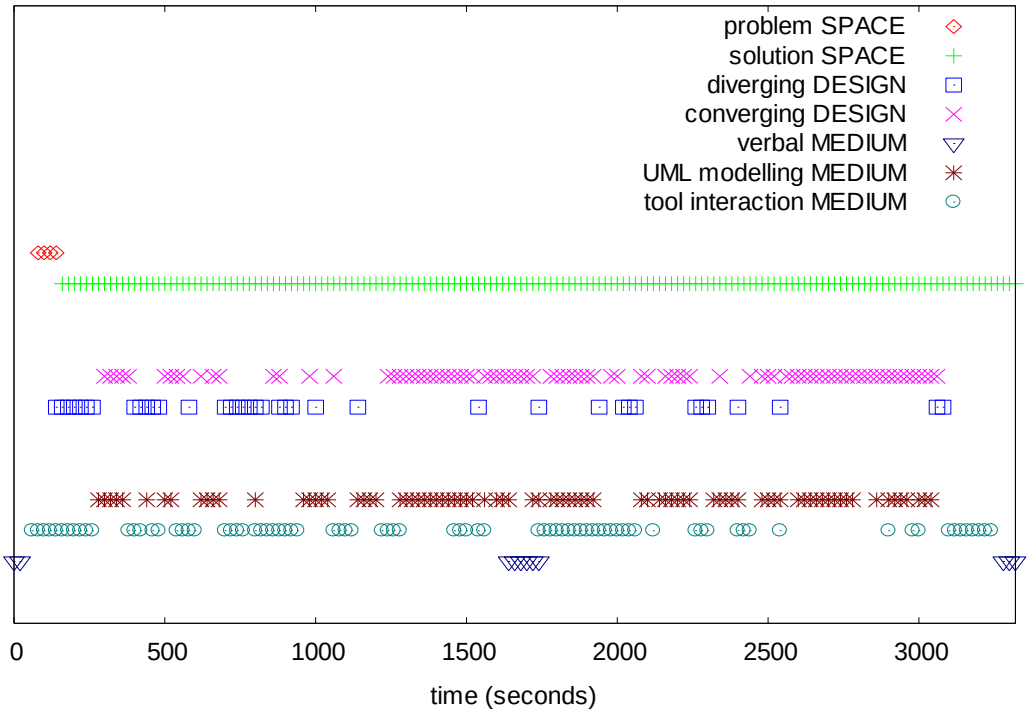

Figure 7.2. Design Modes for Test Episode

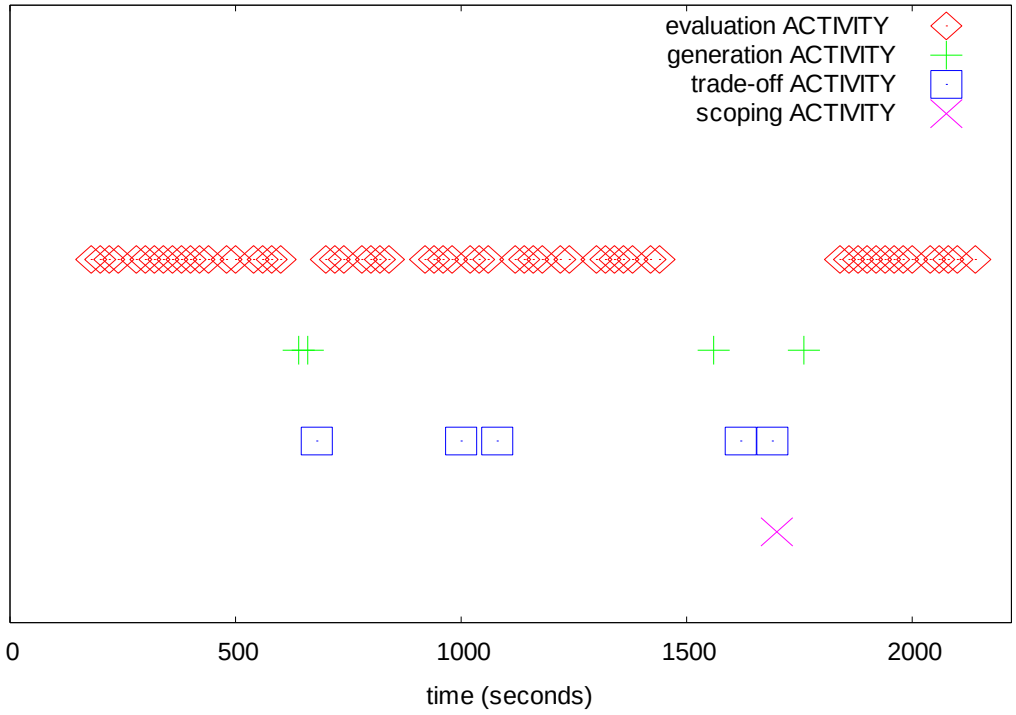

Figure 7.3. Design Activities for Baseline Episode

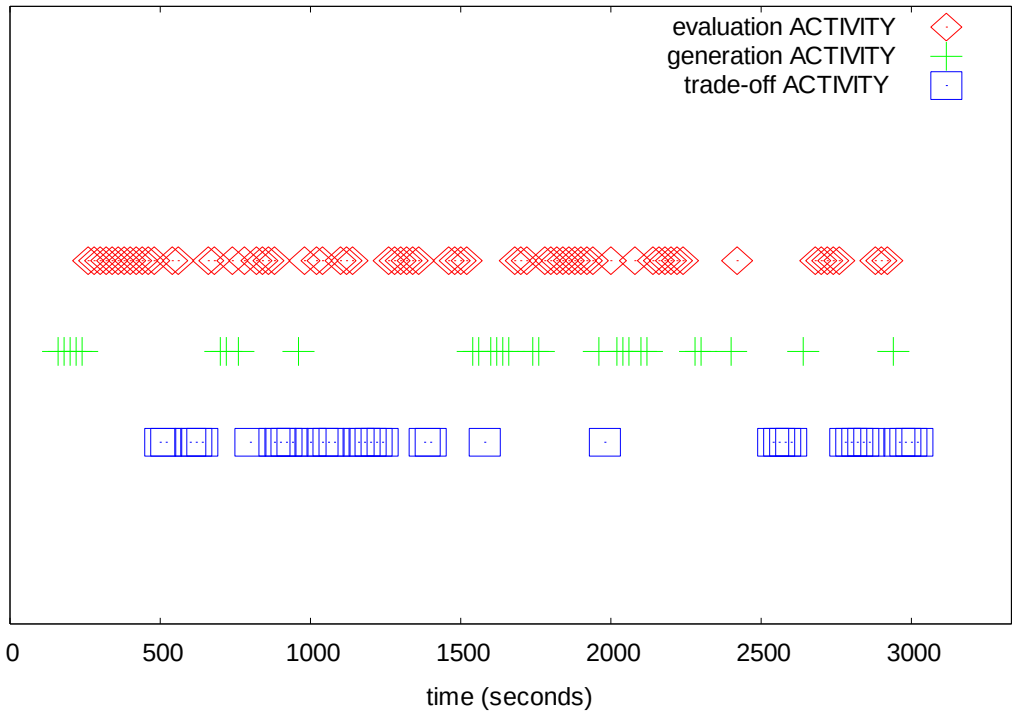

Figure 7.4. Design Activities for Test Episode

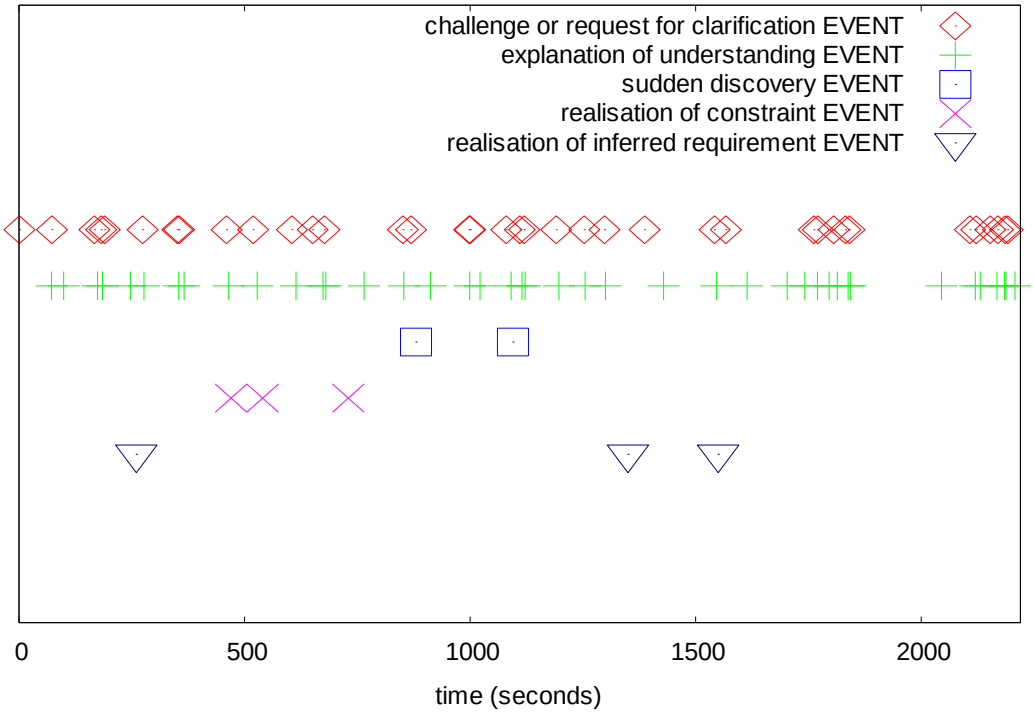

Figure 7.5. Design Events for Baseline Episode

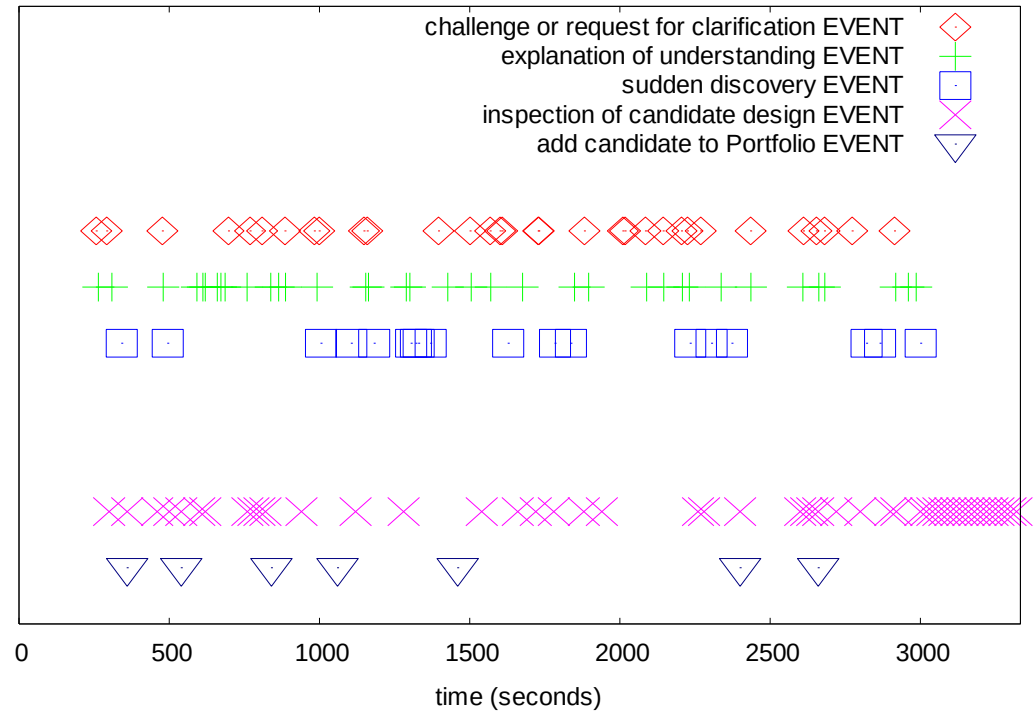

Figure 7.6. Design Events for Test Episode

Table 7.1 Observational Data

ASPECT		OBSERVATION	BASELINE	*Proportion*	TEST	*Proportion*
Duration		Seconds	2212		3323	
		Minutes-Seconds	37-02		56-23	
Mode	Space	Problem	34	0.307	4	0.024
		Solution	77	0.699	158	0.950
	Thrust	Convergent	45	0.406	87	0.523
		Divergent	17	0.153	37	0.222
		Iterations	10		28	
	Medium	Verbal	110	0.994	10	0.060
		Sketching	1	0.090	0	0.000
		Tool Interaction	0	0.000	71	0.000
		UML Class Modelling	0	0.000	85	0.512
Activity		Evaluation	60	0.542	67	0.403
		Generation	4	0.036	28	0.168
		Trading-off	5	0.045	36	0.216
		Scoping	1	0.009	0	0.000
		Reflective Silence	0	0.000	9	0.054
				Ave Freq		*Ave Freq*
Events		Request for clarification	37	59.780	32	103.840
		Explain Understanding	41	530950	37	89.910
		Sudden Discovery	2	1106.000	18	184.610
		Constraint Realisation	3	737.330	0	
		Inferred Requirement	3	737.330	0	
		Inspect Candidate	0		30	110.770
		Add to Portfolio	0		7	474.710

In table 7.1, where proportions are reported for episode modes and activities, these relate to the proportion of the mode or activity as a part of the total duration of the episode. Where average frequencies are reported for episode events, the average frequency in seconds is the episode duration divided by the number of events.

7.5.5 Human Experience

It is interesting to observe that participant reaction to the interactive framework support tool was highly positive overall. After a short period of familiarization, the two participants became fluent in the use of the capabilities provided by the interactive framework support tool. Indeed, by the end of the test design episode, both designers were freely suggesting useful enhancements and extensions to the search-based tool. A full list of the suggested enhancements is available in Appendix C. It is interesting to note that of all the suggested enhancements, the ability to rate the 'quality' of designs and individual classes is considered by the participants to be the most important. Other suggested enhancements include, for example, the ability to find all designs in a population with designer specified groupings of attributes and methods, and the ability to drag and drop attributes or methods from one class to another 'on-the-fly' during search.

7.6 Analysis

With regard to the duration of the two episodes, the participants appeared to respond positively to opportunities presented to explore and exploit designs, resulting in more time spent in the test episode than the manual. Indeed, the test episode would have continued longer had not the interactive framework unfortunately encountered an out-of-memory problem and so was forced to terminate.

With respect to design modes observed, it is clear that iteration between the problem and solution spaces is richer in the manual design episode; less problem / solution iteration is evident in the tool supported episode. This suggests that interactive framework support tends to focus the designers on the solution space. The design thrust of the manual design episode is essentially convergent whereas the tool supported episode shows more divergence and iteration. This may be due to population-based search providing support for extensive exploration. The medium of the manual design episode shows dramatic differences to the medium of the tool supported episode. The

manual design episode is highly verbal, with occasional paper sketching of graphical interfaces but no UML modeling. However, the tool supported design episode is greatly more productive in terms of UML modeling, with over one half of the episode focused on this.

Within the design activities, design evaluation is observed to be the dominant activity in the manual design episode. Indeed, with few candidate software designs being generated, evaluation appears to dominate. However, the interactive framework provides much designer support via generating candidate software designs. Interestingly, tool support also provides opportunities for both quantitative evaluation as well as trade-off evaluation. Trade-off evaluation appears to be a difficult activity in the manual design episode as it requires designers to remember many designs for comparison. Conversely, in the tool supported episode, a design portfolio is provided which greatly assists trade-off evaluation – a significant benefit of interactive framework support. Furthermore, it is observed that colourful visualization of UML class designs enables periods of cognitive reflection. In fact, nine reflective periods of silence were observed in the tool supported episode whereas none were observed in the manual design episode.

Regarding design events, designers were observed to make more requests for clarification at a greater frequency in the manual design episode. In addition, a greater number of verbal explanations of understanding were observed in the manual design episode. This is consistent with the highly verbal medium in which the manual design episode is conducted. Conversely, requests for clarification and explanations were less abundant in the tool supported episode; it seems likely that this is due to the visualizations of candidate software designs that promoted shared understanding of the designs. It is significant that the number of sudden design discovery events were observed to be higher in the tool supported episode (18) than in the manual design episode (2). This finding appears to be consistent with the nine periods of reflective silence observed in tool supported episode. It seems likely that rich generation of alternative candidate designs, when combined with opportunities for visual reflection, affords more opportunities for moments of sudden design discovery. It is also significant that in the tool supported episode, 30 unique software designs were inspected by the participants; i.e. a candidate design was inspected roughly once every two minutes. The designers having been stimulated by the visualization on the UML

designs, 7 were added to the interactive framework portfolio. Lastly, it was observed that the participants expressed the wish to more fully express their opinions on the 'quality' or 'appearance' of the software designs, and elected to award 'star ratings' to designs. Although the interactive framework recorded the star rating of a design textually, it was incapable of exploiting this information during the design episode.

7.7 Threats to Validity

Two designers have been observed in the course of this empirical investigation. While a greater number of designers would add weight to the investigation, this situation reflected the reality of the software development team under study. Moreover, the two designers are representative of some section of the software engineering design community where empirical investigations available in the literature are few.

The above analysis of findings must also tempered by the fact that the same problem domain has been used for both episodes. Given that the manual design episode has been conducted firstly, it is possible that the designers will take any acquired knowledge of the problem domain into the second, tool supported episode. Given this learning effect, it might be reasonably expected that the designers would arrive at a greater number of designs in the tool supported episode. However, it is argued that the numbers of trade-off evaluations (36), moments of sudden design discovery (18), candidate design inspections (30) and additions to the portfolio (7) are considerably higher in the tool supported episode, even accounting for any learning effect. To counter any learning effect, it could also be argued that two different design problems might be used in the study. However, ensuring that any two design problems are strictly comparable for the purposes of an empirical study is a hugely difficult, if not impossible, undertaking to achieve.

7.8 Conclusions

Overall, analysis of the observational data reveals that for this small scale empirical investigation, the interactive framework of agent-mediated, search-based support for early lifecycle software design is effective at generating multiple candidate software designs, and highly productive in terms of visual UML class designs. In the manual design episode, as few candidate designs are generated, manual evaluation of candidate designs is the dominant design activity. In contrast, in the search-based tool supported

design episode, (i) generation of candidate designs is more balanced with evaluation, (ii) evaluation is both manual and computer-based and (iii) trade-off analysis is greatly enhanced. Furthermore, visualization of colourful UML class designs, when combined with generation of multiple candidate designs, enables periods of reflection that stimulate opportunities for sudden design discovery.

With respect to the human experience of interactive search support, feedback from the participants suggests that software designers respond positively when presented with opportunities to interactively explore and exploit many candidate software designs via an interactive framework. Indeed, feedback from the participants also indicates that opportunities for qualitative evaluation of what was described as the ‘quality’ or ‘appearance’ of the software designs might enhance the potential of the interactive framework further. Building upon such feedback, investigations into incorporating qualitative evaluation of design elegance are conducted in the next chapter.