Research Article

Empirical Studies for the Assessment of the Effectiveness of Design Patterns in Migration between Software Architectures of Embedded Applications

Farah Lakhani and Michael J. Pont

Embedded Systems Research Group, University of Leicester, Leicester LE1 7RH, UK

Correspondence should be addressed to Farah Lakhani, fnl1@le.ac.uk

Received 30 May 2012; Accepted 26 June 2012

Academic Editors: O. Greevy and B. C. Lai

Copyright © 2012 F. Lakhani and M. J. Pont. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Two main architectures used to develop software for modern embedded applications are “event triggered” (ET) and “time triggered” (TT). ET designs involve creating systems which handle multiple interrupts; by contrast, only one interrupt is ever enabled in a TT design, and this interrupt is usually linked to a timer “Tick.” Although TT architectures are widely used in safety-related designs, they are less familiar to developers of mainstream embedded systems. The work on this research began from the premise that—for a broad class of systems—the use of a TT architecture would improve reliability. The overall goal of the work presented here was to identify ways in which the effort involved in migrating between existing ET architectures and “equivalent” TT architectures could be reduced. The specific goal of the research was to explore whether the use of an appropriate set of design patterns could assist developers who wished to migrate between ET and TT designs. An empirical evaluation of the efficacy of a newly proposed pattern collection is described in this paper. The results of these trials demonstrate that the proposed collection of patterns has the potential to support developers by helping them to take appropriate decisions during the migration process.

1. Introduction

There are two main architectures used to develop software for modern embedded systems; these can be labelled as “event triggered” and “time triggered” [1–4].

Because of their flexibility in design event-triggered (or “ET”) architectures are mostly preferred by many developers while designing embedded applications. ET designs involve creating systems which handle multiple interrupts. For example, interrupts may arise from periodic timer overflows, the arrival of messages on a serial communication bus, the pressing of a switch, the completion of an analogue-to-digital conversion, and so on. To create ET systems, the developer may write code to handle the various events either directly; this will typically involve creating an “interrupt service routine” to deal with each event. Alternatively, the event will be handled slightly less directly through the use of a real-time operating system.

The alternative to an event-triggered architecture is a time-triggered “TT” architecture. When implementing TT

systems, there is only one interrupt enabled. This single interrupt is usually linked to a timer “Tick,” which might occur (for example) every millisecond; this tick, in turn, drives all software activity in the system.

Although TT architectures are widely used in safety-related and safety-critical designs (e.g., in aerospace and medical systems), they are less familiar to developers of mainstream embedded systems. The research project which is reported in this paper began from the premise that—for a broad class of systems—use of a TT (rather than an ET architecture) would improve reliability. The overall goal of the work presented here was to identify ways in which the effort involved in migrating between existing ET architectures and “equivalent” (and effective) TT architectures could be reduced.

It may be tempting to assume that conversion between ET and TT designs will simply involve converting all event-handling software routines into periodic activities. However, the required software changes are—in many cases—much

FIGURE 1: Association map for the PMES collection.

more profound. The specific goal of work presented in this paper was to explore whether the use of an appropriate set of “Design Patterns” could assist developers who wished to migrate between ET and TT designs.

Though they originated in the field of architecture [5, 6], design patterns have since gained considerable significance in various diverse fields such as object-oriented software development [7, 8], telecommunication systems [9, 10], business organization [11], pedagogy [12, 13], interaction design [14], and cognition [15]. In the area of embedded systems, most previous work with the design patterns has focused on the process of system construction, for example, [16–18].

The research presented in this paper has initiated the idea of using design patterns in migration between different software architectures of embedded applications. To further support this idea the research has resulted in the development of a novel collection of design patterns called the PMES collection (patterns for migration of embedded systems). Please see Figure 1 for the association map between different patterns and the Table 1 for the thumbnails of patterns. The main objective of this pattern collection is to support the developers by helping them to take appropriate decisions during the complex process of migration from ET to TT

designs. To the best of the authors’ knowledge, this study represents the first attempt to use design patterns for the purposes of such complex system (architecture) migration.

The PMES collection has evolved in stages, and most of the patterns have been published previously in [19–21]. A complete detail for all the patterns is described in [22]. Rather than repeating the description of these patterns, this paper has a focus on an evaluation of the effectiveness of the collection as a whole. This type of “evaluation research” can be described as an attempt to assess the worth or value of some innovation, service, or approach [23]. The goal of such an evaluation is to answer questions such as “Does the artefact or theory work?” and “How useful is the artefact or theory?” It has been suggested that—in a research context—the most important benefit of evaluation activity is that it offers feedback to the researcher in order to determine whether the problem is well understood, if the assumptions are appropriate, if the quality of the design process is appropriate, and if further refinements are needed [24]. As the use of patterns becomes more widespread, it seems to be an appropriate time to address such issues.

Following this introduction, the rest of the paper is organised as follows: Section 2 presents a background of

TABLE 1: Thumbnails for the patterns in the PMES collection.

Pattern name	Description
Patterns for migration	
Time for TT?	Discusses about when it is appropriate to use TT architecture
Events to time	Discusses what changes are required to be done at high level when changing the system from ET to TT design
TT scheduler	Discuss what a TT scheduler is and its possible types. Also discuss what type of TT scheduler is appropriate in different situations
Cooperative scheduler	Discusses the implementation of time-triggered cooperative scheduler
Hybrid scheduler	Discuss the implementation of time-triggered hybrid scheduler
Preemptive scheduler	Discusses the implementation of time-triggered preemptive scheduler
Choosing task parameters	Discusses the appropriate choice of task parameters such as offset and order of task execution
Buffered output	Discusses how to deal with long-task problem when working with co-operative scheduler
Polled input	Discusses how to deal with external events in time-triggered environment
Critical section	Discusses how to avoid conflicts over shared resources during the execution of critical sections
Resource lock	Discusses the implementation of resource lock in embedded system
Disable timer interrupt	Discusses the simplest way of implementing safe access to shared resources
Priority inheritance protocol	Discusses the implementation of access to shared resources with mutual exclusion and without priority inversion
Improved priority ceiling protocol	Discusses the implementation of access to shared resources with mutual exclusion avoiding priority inversion, deadlock, and blocking chains
Patterns for optimisation	
Balanced system	Discusses the types of jitter encountered in TT architectures and considers how to design systems with minimum levels of jitter
Sandwich delay	Discusses a technique to reduce period jitter in tasks
Single-path delay	Discusses a technique to reduce tasks execution jitter
Take a nap	Discusses a technique to reduce tasks execution jitter and with minimum power consumption
Planned preemption	Discusses a technique to reduce tick jitter in TTH designs
System monitors	Discusses the need for (and implementation of) monitors in the system which keep track of system state and take necessary actions in case of errors
Loop timeout	Discusses the implementation of a software technique to reset systems in case of any unexpected error that could (otherwise) cause the system to “hang”
Watchdog	Discusses the implementation of a hardware technique to reset system in case of an unexpected error
Task guardian	Discusses the need for (and implementation of) shutdown mechanisms for tasks that exceed their predicted worst-case execution time

the research work and the motivation for this paper. Section 3 discusses the methodology adopted for conducting the experiments described in this paper and the constraints involved in experimentation. Sections 4 and 5 describe the experimental studies in detail and present results. Section 6 summarises and discusses the results obtained. Conclusions are presented in Section 7.

2. Background

The adoption of patterns by the software community has been influenced by the need in this community to reuse software. Software developers have a strong tendency to reuse designs that have worked well for them in the past and, as they gain more experience, their repertoire of design experience grows, and they become more proficient. However, this design reuse is usually restricted to personal experience, and there is usually little sharing of design knowledge among developers [25]. The advent of design patterns

offered an opportunity to overcome the inefficiencies and wasted resources of reinvention and to share the collective experience of the software community.

Embedded software development is even more challenging compared with desktop applications because they are characterised by resource constraints such as limited memory, limited power consumption, and timing constraints [26]. Furthermore, unlike most desktop applications, embedded applications run on specific hardware with special purpose RTOS (real-time operating system), schedulers, programming languages, or network protocols such as CAN. Another major difference is in the cross-development environment. Desktop applications are usually developed on the same platform for which they are designed for, whereas embedded applications are built and tested in simulated environments, and the generated executables are then transferred onto the target processor.

As patterns have the ability to capture domain specific information for the benefit of practitioners, they can

play a vital role in reducing the complexities involved in embedded software development. Some of the previously introduced pattern collections for embedded software development are reported in [16, 17, 27, 28]. However, there was a lack of support for design patterns which could assist in system migration. In the research presented in this paper, we have identified the lack of a standard approach in the use of design patterns in the process of migrating between different software architectures of embedded applications. This research is therefore aimed to provide support to the practitioners in this field by introducing a new pattern collection to assist in the migration process from ET to TT designs. The research introduces a new pattern language for system migration. To illustrate the efficacy of this approach the paper presents empirical studies conducted during the research process.

3. Experiment Planning

3.1. Goals. The goal of the studies described here is to explore two specific hypotheses, by means of empirical studies. These hypotheses are as follows.

- (1) Patterns in the PMES collection help experienced developers to choose appropriate TT solutions during the migration process.
- (2) The PMES form helps the pattern users to apply the information presented in a more effective way than “traditional” information sources.

3.2. Design of Experiments. The key ingredients in designing empirical studies for design patterns are the principal research approach (controlled experiment versus field study), the background of the subjects (students versus professionals), the type of software work investigated (design, development, individual/team work, etc.), and the technical conduct of the study [29]. To evaluate the patterns ideally, they should be tested “in the field” in real embedded software development environments where real practitioners are facing the migration challenge with some “live” projects. However, the studies explained in this paper were designed during a three-year research project, and the main constraints involved were limited funds and limited time to finish the entire project. The proposed collection of patterns for this study as stated earlier is intended to support developers and designers of TT- and ET- based embedded applications. However, because of the practical considerations (mentioned above) under which these studies were designed such an ideal situation was impossible to achieve as industrial projects usually take years to complete, and companies have their own reservations about sharing any information while a project is in process. Given these limitations the next available option was to arrange for some controlled laboratory experiments which could be managed conveniently, and these were chosen for the studies discussed here. However, the experiments conducted were designed and planned according to the recommendations for designing empirical studies in software engineering such as

those described in [30] and provides useful input for further research in the industrial context.

3.3. Selection of Subjects. The next important thing was to decide on the subjects, that is, the class of users for which such a collection can provide support. Again, ideally this should include subjects from a broad cross-section of professionals from the embedded software community such as architects/designers and programmers. This is a challenge, however, because involving such a broad range of subjects under the limitations mentioned was nearly impossible. An alternative is to use students as subjects and studies; for example, [31, 32] highlighted the issue of using students as subjects rather than professionals working in the industry. These issues are related to the benefits that researchers gain from empirical studies with students such as to obtain preliminary evidence to confirm or refute hypotheses and benefits to students such as getting better insights on specific industrial problems. Interestingly, the studies encourage the use of students as subjects particularly when some of the students had the same level of skill as the professionals. Therefore, many studies published in literature such as [33–35] has used students as subjects for pattern evaluation.

Within the limited funding constraints it was not possible to involve a huge number of participants in the study given that the participants were paid £20.00 each in appreciation of their time and effort required to conduct the studies. Therefore, for the studies reported in this paper, choosing small “balanced” group of subjects with equal background and capabilities was emphasised. This is also recommended in previous studies such as [36, 37]. To do this, an account was taken of the subject’s prior experience and the courses they had undertaken. Moreover, given the relatively easy access to students in this research environment, M. S. (engineering) and M. S. (computer science) students at the University of Leicester were selected as subjects.

3.4. Task Design and Assessment Methodology. For empirical studies of software engineering there are several dimensions in which the task performed by the experimental subjects may differ; it may be a design or implementation task, from scratch or in maintenance, may be done alone or by a team, may target programs of different size and from different domains, and may employ different types of design patterns [29]. For the studies reported here tasks were designed such that the usability of patterns can be assessed to its maximum. Two aspects were considered: (1) the uniqueness of the design problem that pattern has addressed and (2) the pattern form. It was decided not to use any tasks involved coding, rather the exercises designed were completely analytical, and the aim was to test how patterns in the PMES collection can help designers in taking the appropriate decisions during the migration process.

For a fair and unbiased assessment of the exercises the “blind” methodology discussed in [38] was adopted. In this case a neutral person or a third party which has no direct involvement in the research is hired for the assessment. In this case the third party who analyses the results is

kept unaware as to which sets of results have been subjected to a treatment. For the studies discussed in this paper, some senior members of the research group helped in this regard by offering their time for grading of the exercises solved by the subjects. The assessors were acting as a third party and were not told about the aims of the study, neither were told about which exercise sheets were solved with or without using patterns in order to achieve a completely fair set of results which are completely unbiased. After the results obtained, the data analysis was done by the experimenter himself.

The next two sections will provide full details about the experiments.

4. Experiment 1: Patterns for Experienced Developers

4.1. Aims and Motives. Experienced developers of embedded systems are usually adept in designing applications from scratch and are usually confident about the systems which they have designed with the architectures and tools at their disposal. However, it can be quite a challenge for experienced developers to transform an existing architecture to another especially when they are not too familiar with the architecture that they wish to transform. This first experiment was designed for this aforementioned situation and investigates the following hypothesis.

Hypothesis H1. Patterns in the PMES collection help experienced developers to choose appropriate TT solutions during the migration process.

4.2. Preparation of Exercise. An exercise was prepared based on the details of three different embedded applications that were designed in the ET architecture and needed to be migrated to the TT architecture. The exercise was completely analytical with no coding requirements and the applications given in the exercise are summarised below:

- (i) traffic control system (TCS): an application based on the controller for the traffic lights and pedestrian crossing lights used at a typical crossroads in the UK,
- (ii) fast fourier transform on ADC samples (FFT/ADC): this application is based on two main activities: data sampling using ADC and time-frequency conversion using fast-fourier transform (FFT),
- (iii) data acquisition system (DAQ): a control application used to sample data from an analogue-to-digital converter (ADC), translate the sampled value to an appropriate string of characters, and display this value and the elapsed time since the microcontroller was last reset on to the screen on user request.

The author herself has already worked on the design/development of all the above applications and was fully aware of the design constraints involved in each applications. All the applications differed in their design, difficulty level and complexity where complexity is measured in terms of the

TABLE 2: Difficulty levels of exercise.

Application	Lines of code	Complexity
TCS	3400	High
ADC/FFT	1509	Medium
DAQ	600	Low

TABLE 3: Group structure for the study.

Group ID	Members	Patterns	Marks in PES-I
PG1	2	✓	In the range of 90%
NPG1	2	×	
PG2	2	✓	In the range of 70%
NPG2	2	×	

number of lines of code (LOC) in each system (see Table 2). The reason for choosing systems of varied complexity is to test the usability of patterns more broadly.

The details on the existing architecture of each of the applications were described in the exercise. After the related details of each system, the subjects were asked to choose an appropriate TT architecture (to test the migration patterns in the PMES collection) and decide on what measures they can take in order to make the achieved TT system more reliable and predictable (to test the optimisation patterns in the PMES collection).

4.3. Management of Subjects. For this experiment, it was decided to involve M. S. (engineering) students who had already taken a 10-week, one-semester module in programming embedded systems (PES-I) and were doing the extended module PES-II during this study. Therefore, the subjects involved in this experiment had sound intermediary knowledge and prior experience of design and programming embedded applications. An email was circulated to the group of students enrolled on the PES-II module inviting them to participate in the research study. The subjects were not told about the actual theme and purpose of the study to keep the results unbiased, and this also helped to eliminate the possibility of the subjects' behaviour being influenced by knowledge of the experimenter's expectation. This is known as the hawthorne effect in the literature [38, 39]. The test subjects then were only informed of the study's objectives and motives at the end of the experiment.

From a list of fifteen volunteers, eight subjects were selected for the study such that their mark history in PES-I was in 90–70% range. The participants were organised as PG (pattern group, that is, the group members were given with the PMES pattern collection during the exercise) and NPG (nonpattern group, that is, the group members were not given with the PMES pattern collection during the exercise). The top scoring members for the PES-I module were put in pairs PG1 and NPG1 and the four who scored around the 70% mark were also put in pairs PG2 and NPG2 as shown in Table 3.

The “N” in these groupings refers to the pairs that were not given patterns to help them in the exercises. Therefore,

TABLE 4: Performance of groups.

Groups	TCS	DAQ	FFT/ADC
PG1	66%	100%	75%
NPG1	40%	100%	75%
PG2	90%	100%	95%
NPG2	50%	75%	60%

the NPG was expected to use their prior experience in embedded system design to solve the exercises during the study and the PG groups were provided with the PMES collection of patterns. The purpose of such an organisation of participants is to achieve fair results at the end. As participants in PG1 and NPG1 bear almost same level of skills and well-matched academic backgrounds (reflected in their results 90% range) and so is considered for PG2 and NPG2 (previous results in 70% range).

4.4. Procedure. The experiment was conducted in the Electrical Teaching Laboratory (ETL) of the Engineering Department in the University of Leicester. All of the groups were called to attend the study on the same day and were seated far apart from each other (to reduce the possibility of interaction among groups during the study). During the study, all the groups were constantly under observation (by the author herself) to make sure that they were completely focused on the study and not wasting time as the measurement of time was vital data for this study. This observation was as unobtrusive as possible so that those being observed were not distracted from the task at hand as noted by Seaman [40] in her paper on empirical studies of software engineering. This was to help ensure that the observed behaviour was normal, that is, what usually happens in the environment being observed, and is not affected by the presence of the observer. Each of the paired groups was asked to take strict notice of the start time and the finish time they spent on each system. There was no time limit restriction to finish the exercise, and the groups were free to spend as much time as they wanted. The exercise sheets were collected back when all groups had completed the tasks and were forwarded to the assessors.

4.5. Data Analysis. For the purpose of data analysis, two variables were important for this study: the performance of the subjects and the time spent on finishing the exercise.

After the assessment was done, the performance of each group was calculated using

$$\text{Performance} = \frac{\text{marks obtained}}{\text{maximum marks}} * 100. \quad (1)$$

The results obtained for each individual task in the exercise for each of the group are summarised in Table 4 and plotted in Figure 2.

The results reveal some interesting facts. For example, in the case of the DAQ application there is no high difference in the performance between groups PG and NPG except for NPG2, and even they obtained a high score(>70%).

FIGURE 2: Performance of groups.

This suggests that patterns did not have a significant impact in helping the subjects working on systems with lower complexity as DAQ is at lowest level of complexity in this exercise (please see Table 3). On the other hand there is a considerable difference in the results for PG and NPG groups for the TCS system which at the highest level of complexity indicated that patterns helped the subjects in understanding more complex architectures.

In software engineering experiments, one important characteristic of the sample data is its mean which is represented by $\bar{x}$. A better understanding of the results can be achieved by using some measure of the dispersion of the population data [30]. The most commonly used measure is the variance of the population which is represented by " s^2 ." The positive square root of variance called standard deviation " s " can also provide useful analysis for the data in hand. For a small sample size such as $n \leq 30$ the mean $\bar{x}$, variance s^2 and standard deviation s are usually calculated as follows:

$$\text{Mean} = \bar{x} = \sum \frac{x_i}{n}, \quad (2)$$

$$\text{Variance} = s^2 = \sum \frac{(x - \bar{x})^2}{n - 1}, \quad (3)$$

$$\text{Standard deviation} = s = \left[\sum \frac{(x - \bar{x})^2}{n - 1} \right]^{1/2}. \quad (4)$$

For the results obtained shown in Table 4, mean, variance, and standard deviation are calculated and plotted in Figure 3.

Results shown in Figure 3 above indicate the performance benefits of using patterns in both PG groups that is, their performance is above the mean value. Overall the pattern user groups performed better than the groups that worked without patterns.

Regarding the validity of the statistical hypotheses in software engineering experiments Juristo and Moreno [30] have described some decision rules. According to them the differences between the means of the sample data can help

FIGURE 3: Comparison of group performance with respect to mean and standard deviation.

TABLE 5: Calculations of the difference in means of PG and NPG groups.

	PG	NPG
Performance of groups in (%)	80	72
	95	62
Sum	$\sum PG = 175$	$\sum NPG = 134$
Mean	$PG' = 87.5$	$NPG' = 67$
Difference of mean	$PG' - NPG' = 20.5$	

in accepting or rejecting a hypothesis and analysing the significance of it. A hypothesis is normally rejected if the differences between the means (for two alternatives) is either zero or a value less than 5%. Alternatively, higher values of the difference between the means indicate that the hypothesis is acceptable and proves the validity and significance of the stated hypothesis. For the validity of the hypothesis H1 the mean of the performance of PG and NPG groups and the difference in mean is calculated and is shown in Table 5.

The high value of difference of mean indicated that hypothesis H1 is acceptable for the experiment described above.

For the time measurements, the total time spent by the groups on the whole exercise was determined from the recorded start and finishing times spent on each application. In order to further analyse the results the overall time spent (in minutes) is plotted graphically and shown in Figure 4.

It is quite interesting to note that the PG2 group spent overall more time than PG1, and this was because this group has spent more time to familiarise themselves with the information provided in the patterns (as noted by the observer during the experiment). To look back at the performance of PG1 (the 90% scorers in the PES-1 module) and PG2 (the 70% scorers in the PES-1 module) PG2 performed extremely well in the exercise (please see Table 4). This implies that expert knowledge and availability of patterns is not enough, but lesser expertise plus a more thorough study of patterns can be very productive.

FIGURE 4: Time taken by each group to complete the exercise.

5. Experiment 2: Patterns versus Alternative Resources

5.1. Aims and Motives. At one level, a pattern is simply a structured document which formalises the relationship between a nontrivial problem and a nonobvious solution. It is implicit in much of the work on design patterns that this “pattern form” offers advantages over traditional ways of representing this type of information (e.g., in a textbook). Different authors have their own ways of documenting patterns. However, certain pattern forms have become more established than others, for example *Alexandrian form* used by Alexander in [5], *GoF form* used to write famous software patterns for object-oriented software by [7], *POSA form* used to write patterns for software architecture [41] and *PTTES form* used by Michael Pont to write patterns for time-triggered embedded systems [17].

For the PMES collection we have followed the PTTES form because of its relevancy with the context. All the patterns in the PMES collection have information organised in the layout shown in Box 1.

For this experiment the specific hypothesis under test was the following.

Hypothesis H2. The PMES form helps the pattern user to apply the information presented in a more effective way than “traditional” information sources.

5.2. Management of Subjects. One of the potential users of the proposed pattern collection is new graduates of computer science who are about to start their career in the area of embedded systems development. Graduates in computer science have already taken essential courses in software development/programming and have a general set

Pattern Name**Context**

This describes the situation for which the pattern is applicable.

Problem

This is the statement and description of the problem for which the pattern is documented.

Solution

This describes the core of the solution to the problem.

Related patterns and alternative solutions

This section contains information regarding related patterns that may be affected by the application of the solution described in the pattern and other possible alternative solutions.

Reliability and safety issues

This gives reliability information specific to the pattern.

Examples

This includes relevant code examples.

Overall strengths and weaknesses

☺ Positive effects

☹ Negative effects

Box 1: Pattern form used for documenting the patterns in the PMES collection.

of skills which are attractive to companies working in the business of embedded systems. In applying and extending their skills to new engineering problems in a professional environment, such graduates would rely on resources such as books, manuals, research papers, and other online material to complement their learnt knowledge and tacit skills to help solve the problems.

For the selection of subjects in this experiment, an email was sent to the group of the M. S. (computer science) students of The University of Leicester, and out of the list of volunteers 10 high performing students were chosen whose academic performance was comparable, and most of them were distinction holders in their previous semesters. In order to fill in any knowledge gaps the participants may have had on embedded systems, an introductory tutorial was prepared (by the author herself) and emailed to all the participants a week before the study. The introductory tutorial covered the essential concepts of embedded systems and the terms that would be used in the exercise.

For this experiment participants worked individually instead of in groups. The reason for this choice was to avoid differences in the ways the supplementary material would have been used and digested. Out of the ten candidates, five individuals were randomly selected to work with patterns and were given the identification PP1 to PP5 (PP refers to pattern participant), and the five that worked without patterns were given the identification from NPP1 to NPP5 (NPP refers to nonpattern participant). This distribution is shown in Table 6.

5.3. Preparation of the Exercise. For this study an exercise was prepared based on smaller tasks with the aim of testing the hypothesis H2. There were 4 different tasks in the exercise, and each was designed to test a different pattern and with a consideration that the relevant information must be

TABLE 6: Participants in the study.

Pattern participants	PP1	PP2	PP3	PP4	PP5
Non-pattern participants	NPP1	NPP2	NPP3	NPP4	NPP5

TABLE 7: Details of the tasks given in the exercise.

	Relevant patterns	Relevant resources
Task 1	Events to time	[1, 2, 4, 45]
Task 2	Single-path delay Take a nap	[42, 43, 46, 47]
Task 3	Loop timeout	[48]
Task 4	Choosing task parameters	[49, 50]

available through other supplementary resources provided to the nonpattern participants to compare results. For example, Task 2 in the given exercise was related to a programming technique which can allow programmers to make the worst-case execution time (WCET) of a task equal to the best-case execution time (BCET) in order to improve code predictability. In the newly proposed PMES collection, pattern single-path delay under the umbrella of abstract pattern balanced system has discussed this technique. Originally, this technique is proposed by Peter Puschner and is discussed in some of his papers [42, 43]. More details about each task are given in Table 7.

5.4. Procedure. At the beginning of the experiment all the participants were given with the exercise and resources to help them to solve the exercise. The pattern participants (PP1 to PP5) were given with the patterns in the PMES collection along with other supplementary material, that is, the relevant research papers and books. The nonpattern-participants (NPP1 to NPP5) were provided with the same

TABLE 8: Performance of students in Experiment 2.

	PP1	PP2	PP3	PP4	PP5	NPP1	NPP2	NPP3	NPP4	NPP5
Pre-Test	60%	80%	60%	70%	80%	60%	80%	60%	60%	80%
Real	67.5%	90%	70%	65%	70%	37.5%	27.5%	55%	17.5%	30%

relevant research papers and books only (books and papers were same as given to the pattern participants). A pretest assessment was carried out to make sure that all subjects had gone through the introductory tutorial provided before the study, and have a basic understanding of the embedded system concepts required to solve the exercise. To ensure this all the subjects were asked to write a short paragraph about their understanding on ET and TT architectures.

After the pretest, participants were given the actual exercise and relevant research papers. Participants from PP1 to PP5 were also given with copies of the PMES pattern collection, and they were given the option to use them if they wished. However, it was interesting to note during the study that all participants who were provided with the pattern collection preferred to use them against papers/books as their first choice. An obvious reason is the clearer and precise impression of a pattern as a training document compared with its more general description in research papers and books. During the study, as part of capturing the essential data in this study, all the participants were asked to note down the start time and finish time for each task. On completion of the exercise, subjects were asked to complete questionnaires designed to obtain their feedback on the exercise. Different questionnaires were designed for the PP and the NPP subjects.

5.5. Data Analysis. For the data analysis, individual performance of the participants in the pretest which is based on the tutorial and in the actual exercise is calculated using (1), and the results are shown in Table 8 and Figure 5. In the pretest, participants scored between 60% and 80%. This score was random between PP and NPP participants indicating that the average knowledge of basic embedded systems concepts among all the participants was nearly equal. However, the results of the actual exercise show a clear difference in performance of the participants who worked with patterns compared with those who worked with other supplementary materials provided to them.

Results shown in Figure 5 clearly indicate the difference of performance for PP and NPP participants. For further data analysis, same methodology is adopted as discussed in Section 4.5, and the values of mean, variance and standard deviations are calculated using (2)–(4) and plotted as shown in Figure 6.

The performance of all the PPs is higher than the mean of the overall result.

The total time taken by each participant to solve the complete exercise was calculated and is shown in Figure 7. The graph clearly indicates the difference in the time taken by the PP and NPP participants to complete the exercise.

FIGURE 5: Performance of participants.

FIGURE 6: Comparison of the performance of participants with respect to mean and standard deviation.

Results shown in Figures 6–7 indicated that the subjects provided with the pattern-based representation of the material generally managed to absorb the knowledge and apply it more quickly than those who have got the equivalent material in the form of research papers and books. This implies that patterns also provide a way of representing information in a way which is easier to digest and understand. To investigate the acceptance level and significance of the hypothesis H2 the differences of the mean are calculated as shown in Table 9.

The difference of mean's is a high value indicating that the hypothesis H2 is highly acceptable.

FIGURE 7: Time taken by the participants to complete the exercise.

TABLE 9: Calculations of the differences of mean for experiment 2.

	PP	NPP
Performance of participants in (%)	67.5	37.5
	90	27.5
	70	55
	65	17.5
	70	30
Sum	$\sum PP = 362.5$	$\sum NPP = 167.5$
Mean	$PP' = 72.5$	$NPP' = 33.5$
Difference of mean	$PP' - NPP' = 39$	

5.6. Feedback of Subjects. Some important results were also extracted from the questionnaires given to all the participants at the end of the study. This section will discuss the results obtained.

5.6.1. Difficulty Level of the Exercise. In one of the questions the participants were asked “How difficult did the exercise appear to you given there was a requirement to give extra effort to find the relevant information from the additional resource material provided?” A difficulty level (based on a Likert-type scale (a Likert scale is a psychometric scale commonly involved in research that employs questionnaires. It is the most widely used approach to scaling responses in survey research. The scale is named after its inventor, psychologist Rensis Likert (Wikipedia))) was defined in the questionnaire going from 1 to 4 where 1 = too easy, 2 = easy, 3 = difficult and 4 = too difficult. The results are depicted in Figure 8.

The results showed that for PP’s the overall exercise was easy to solve compared with NPP’s who were struggling to find the relevant information from books and research papers.

5.6.2. Value of the Material Provided. To obtain feedback about the documents provided another question asked in the

FIGURE 8: Feedback from the participants regarding the difficulty level of the exercise.

FIGURE 9: Feedback of the participants about the value of the material provided.

questionnaire “Did you find the given documents helpful to solve the tasks given in the exercise?” The results are shown in Figure 9. The participants were again given the following four options: (1 to 4) where 1 = not helpful, 2 = a little bit helpful, 3 = helpful, and 4 = very helpful, and invited to select one.

Participants who worked with patterns found these documents very helpful (average rating = 3.6) while participants provided with other documents felt that they were only a little bit helpful (average rating = 2.4). These results also reflected the significance of patterns as documents because of their structural simplicity and the distribution of information into clear sections of problem, context, and solution.

5.6.3. Feedback about Patterns. There are certain elements in a pattern which play an important role in the success of a pattern even, for example, the name of the pattern that is an important consideration. If the name encodes the patterns’ meaning well, a designer can more easily find a suitable

FIGURE 10: Feedback about pattern names and solution (1–5) 1: Lowest, 5: Highest.

pattern in an unfamiliar pattern language [44]. In the PMES collection most of the pattern names are quite simple such as Time for TT? Events to Time and TT Scheduler, and they give a pretty clear indication of the purpose for which they are designed. Some pattern names however are chosen to describe their function by analogy such as Sandwich Delay, Single Path Delay, and Take a Nap, and these names can be problematic to inexperienced designers who may be unfamiliar with the domain.

Also, at the heart of a pattern is the solution described in it. It was felt important to get feedback of individuals about the clarity of different sections of each individual pattern in the given pattern collection used during the study. Therefore, one of the questions included in the questionnaire related to rate the individual sections of each pattern. The participants were invited to rate the names of the patterns and solution described in terms of the clarity they gave in representing the actual function of the patterns. The clarity was quantified using a five-point Likert scale from 1 to 5 where 1 = lowest and 5 = highest. The results obtained are shown in Figure 10 gained from the participants' feedback.

The pattern Choosing Task Parameters received the highest rating in terms of the clarity of the name whilst Take a Nap is least understandable because it requires one to unpack its analogical meaning.

For the clarity of the solution, the pattern Events to Time appeared as the most highly rated pattern (average 4.4) while Choosing Task Parameters as lowest rated (average 3.2). This information provides useful feedback for making improvements to patterns.

As a preliminary evaluation all published patterns passed through “*Shepherding*” process. This is a process of bringing improvements in a pattern through the critical review of an expert various PLoP (numerous international conferences on pattern languages of programming “PLoP” are organised by the patterns community (e.g. the Hillside Group <http://hillside.net/>) every year as a forum to discuss the latest patterns and pattern languages. As part of the refinement process, pattern languages and (individual patterns) are critically reviewed by experts at PLoP events) conferences.

The results indicated that patterns that have already passed through “*Shepherding*” and have been assessed at “*Writer’s Workshops*” at various PLoP conferences are better understood by users. For example the pattern Choosing Task Parameters is documented in the later stages of the research and has not passed through any PLoP conferences yet, and this could be one of the reasons of its lowest rating.

6. Discussion

6.1. Summary of the Experimental Results. All newly proposed patterns must pass through a process of evaluation so to assess their quality and efficacy for the domain they are designed for. In this regard, two different empirical studies were conducted to assess the novel pattern collection generated as an outcome of this research. Literature provides evidence that evaluation studies for pattern collection are highly dependent on characteristics of application domain and practical constraints faced while conducting such studies. In the light of such evidences, the studies presented in this chapter were designed with the aim of gaining maximum useful results within the presence of constraints.

Experiment 1 discussed in this chapter was designed to test the usability of the PMES collection for reasonably experienced designers/developers of embedded applications. The results have shown that a team of experienced embedded software designers can enhance their insights in software development with the use of patterns during the migration process from ET to TT designs for complex embedded applications. Groups that worked with patterns exhibited better performance in the exercises given to them during the study compared with those who were completely dependent on their previous skills and experience. Figure 11 presents a summary of the results that were obtained relating to Experiment 1.

The graph indicates the following.

- (i) The performance of the PG groups is much better as compared with the NPG groups.
- (ii) The performance of PG2 is even better than PG1 possibly because PG2 spent more time on understanding the patterns.
- (iii) The time taken by PG2 and NPG1 is almost equal but the performance difference of the two groups is higher with, PG2 producing a 20% higher performance than NPG1 in the same time span.

Comparing again the results for the PG and NPG groups, PG1 performed better and took less time compared with NPG1 and PG2 performed considerably better but took more time compared with NPG2. The performance of the PG groups is better in all cases indicating that patterns played an effective role but at this stage it is not sufficiently clear whether the use of patterns increases or decreases the time involved. For this, a further assessment with a larger sample size would be required to draw any firm conclusions. There may be an additional contribution to the time taken by the pattern users on familiarising themselves with the actual patterns so further work could identify any such contribution

FIGURE 11: Summary of results (Experiment 1).

by providing training beforehand on the use of patterns. As mentioned earlier budgetary constraints limited the sample size in this research, and prior training brings its own problems in that there is a risk that the subjects might share information before the study. This latter issue of external validation in controlled experiments limiting the ability to generalise the experimental results has been discussed by Prechelt et al. [33].

Experiment 2 was designed to test the usability of patterns as documents compared with other supplementary materials which inexperienced developers of embedded applications refer to early in their career. The results showed that patterns are a useful replacement to other supplementary materials such as research papers, books, and reference manuals. Patterns presented as structured and more concise documents proved to be a more digestible source of information providing both quick access to the information required and an overall time saving to completion of the exercises. It was observed that participants provided with supplementary material other than patterns experienced a greater difficulty in understanding the tasks. They also spent more time completing them and showed a lower performance in the tasks.

To scrutinise this data further a difficulty index for each individual participant is calculated and defined in (5) below:

$$\text{Difficulty index} = \frac{\text{observed difficulty} * \text{time taken}}{\text{maximum absolute difficulty}}. \quad (5)$$

The maximum absolute difficulty (shown in (6)) is the product of the maximum difficulty level (scale set from 1–4 where 4 = too difficult) and maximum time taken by any of the participant to complete the exercise (observed value = 140 minutes) as follows:

$$\text{Maximum absolute difficulty} = \text{maximum difficulty} * \text{time}. \quad (6)$$

The difficulty index for each participant was calculated and plotted against performance in order to analyse the overall

FIGURE 12: Summary of results (Experiment 2).

impact of using patterns during the study. The results are shown in Figure 12. The trend of the results suggests that individuals who worked with patterns during the study showed much better performance in less time (lower values of difficulty index).

On the other hand, participants who used other supplementary material during the exercise were unable to perform well and worked for a longer time (higher values of difficulty index) as they had the extra demand of extracting related information from the material provided.

In general, the results obtained from both the studies are useful because they helped in the evaluation of the patterns. For example, the better performance of pattern users in both the studies indicated that the proposed collection of patterns is “effective” in the sense that it helped the users to identify the root causes of the problems (given in different tasks in the exercise) and address them with the solution recommended in the patterns.

However, the studies conducted are not free from problems of interpretation—the main one being the small sample size of the subjects. However they do represent a first step towards ascertaining the benefits of applying patterns related to the problems of migration of architecture for complex embedded applications.

6.2. Wider Applications of the PMES Collection. Patterns in the PMES collection are applicable to a wide range of systems. Current research has a focus on industrial applications of this collection.

For example, the advent of new industry standards such as ISO 26262 for functional safety of passenger cars presents many new challenges to organizations in the automotive sector. Some of the principles defined for the software architectural design in ISO 26262 are hierarchical structure of software components, restricted size of software components, restricted coupling between components, appropriate

scheduling properties, and restricted use of interrupts. Some mechanisms are recommended for error detection and error handling at the software architectural level. We believe that the PMES collection has the potential to help developers in the automotive sector to meet some of the challenges involved in meeting ISO 26262 requirements.

We are planning to conduct an industrial survey to get feedback on patterns from professionals in this industry. We are hopeful that this survey will help us to further assess, refine, and extend this pattern collection.

7. Conclusions

It has been suggested that using design patterns has proved beneficial in various fields; the research presented in this paper attempted to assess the usefulness of patterns when changing the software architecture used in an embedded system.

The first experiment described in this paper was designed to test the effectiveness of patterns when employed by experienced developers/designers. The results suggested that appropriate patterns can help developers to take appropriate decisions during the migration of complex embedded applications, and to produce effective systems after migration.

The second experiment was designed for inexperienced developers. The results suggest that appropriate patterns provide an effective way of representing information which is easier to understand, digest, and implement when compared to some other resources, such as research papers and books.

The results presented in this paper are obtained from pilot studies based on a small number of subjects; however, the results are informative and of practical relevance.

References

- [1] H. Kopetz, "Event-triggered versus time-triggered real-time systems," in *Proceedings of the Workshop on Operating Systems of the 90s and Beyond*, 1991.
- [2] H. Kopetz, *Real-Time Systems Design Principles For Distributed Embedded Applications*, Kluwer Academic Publishers, 1997.
- [3] R. Obermaisser, *Event-Triggered and Time-Triggered Control Paradigms*, Springer, 2005.
- [4] F. Scheler and W. Schroder-Preikschat, "Time-triggered versus event-triggered: a matter of configuration?" in *Proceedings of the MMB GI/ITG Workshop on Non-Functional Properties of Embedded Systems*, VDE, 2006.
- [5] C. Alexander, S. Ishikawa, and M. Silverstein, *A Pattern Language*, Oxford University Press, 1977.
- [6] C. Alexander, *The Timeless Way of Building*, Oxford University Press, 1979.
- [7] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison Wesley, 1995.
- [8] R. Hanmer, Ed., *Software Patterns for Reusable Design*, 2009.
- [9] L. Rising, Ed., *Design Patterns in Communication Software*, Cambridge University Press, 2001.
- [10] R. Hanmer, *Patterns for Fault-Tolerant Software*, John Wiley & Sons, 2007.
- [11] M. L. Manns and L. Rising, *Fearless Change: Patterns For Introducing New Ideas*, Addison Wesley, 2004.
- [12] J. Bergin, "Fourteen pedagogical patterns," in *Proceedings of the 5th European Conference on Pattern Languages of Programming*, 2000.
- [13] A. Fricke and M. Volter, "Seminars: a pedagogical pattern language about teaching seminars effectively," in *Proceedings of the 5th European Conference on Pattern Languages of Programs*, 2000.
- [14] J. O. Borchers, "Designing interactive music systems: a pattern approach," in *Proceedings of the 8th International Conference on Human Computer Interaction: Ergonomics and User Interfaces*, vol. 1, pp. 276–280, 1999.
- [15] K. M. Gardner, A. Rush, M. Crist, R. Konitzer, and B. Teegarden, *Cognitive Patterns*, Cambridge University Press, 1998.
- [16] M. Bottomley, "A pattern language for simple embedded systems," *Proceedings of the Pattern Languages of Programming (PLOP '99)*, Chicago, Ill, USA, 1999.
- [17] M. J. Pont, *Patterns for Time-Triggered Embedded Systems*, ACM Press, 2001.
- [18] V. P. Eloranta, J. Koski, M. Leppänen, and V. Reijonen, "Software architecture patterns for distributed embedded control systems," in *Proceedings of the 14th European Conference on Pattern Languages of Programming (EuroPLOP '09)*, A. Kelly and M. Weiss, Eds., CEUR Workshop Proceedings. 566, Irsee, Germany, 2009.
- [19] F. Lakhani, A. Das, and M. J. Pont, "Towards a pattern language which supports migration of systems from event-triggered pre-emptive to time-triggered co-operative software architectures," in *Proceedings of the 14th European Conference on Pattern Languages of Programming (EuroPLOP '09)*, Irsee, Germany, 2009.
- [20] F. Lakhani, A. Das, and M. J. Pont, "Creating systems with predictable patterns of behaviour: Migrating systems from event-triggered to time-triggered software architectures," in *Proceedings of the 15th European Conference on Pattern Languages of Programming (EuroPLOP '10)*, Irsee, Germany, 2010.
- [21] F. Lakhani and M. J. Pont, "Using design patterns to support migration between different system architectures," in *Proceedings of the 5th International Conference on System of Systems Engineering (SoSE '10)*, Loughborough, UK, June 2010.
- [22] F. Lakhani, H. Wang, and M. J. Pont, "Supporting the migration between event-triggered and time-triggered software architectures: a small pattern collection intended for use by the developers of reliable embedded system," Technical Report ESRG, 2011.
- [23] C. Robson, *Real World Research: A Resource For Social Scientist and Practitioner*, Blackwell Publishers, 2002.
- [24] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.
- [25] K. Beck, J. O. Coplien, R. Crocker et al., "Industrial experience with design patterns," in *Proceedings of the 18th International Conference on Software Engineering*, pp. 103–114, Berlin, Germany, March 1996.
- [26] C. G. Buttazzo, *Hard Real-Time Computing Systems Predictable Scheduling Algorithms and Applications*, Kluwer Academic Publishers, 1997.
- [27] W. Herzner, W. Kubinger, and M. Gruber, "Triple-T (Time-Triggered Transmission)" A system of patterns for reliable communication in hard real-time systems," in *Proceedings of the 5th European Conference on Pattern Languages of Programming (EuroPLOP '05)*, Irsee, Germany, 2005.

- [28] V. P. Eloranta, J. Koski, M. Leppanen, and V. Reijonen, "A pattern language for distributed machine control system," Tech. Rep. 9, Tampere University of Technology, Department of Software Systems, 2010.
- [29] L. Prechelt and B. Unger, "A series of controlled experiments on design patterns: methodology and results," in *Proceedings of Softwaretechnik*, Softwaretechnik-Trends, 1998.
- [30] N. Juristo and A. M. Morena, *Basics of Software Engineering Experimentation*, Kluwer Academic Publishers, 2001.
- [31] J. Carver, L. Jaccheri, S. Morasca, and F. Shull, "Issues in using students in empirical studies in software engineering education," in *Proceedings of the 9th International Symposium on Software Metrics*, 2003.
- [32] E. Arisholm and D. I. K. Sjøberg, "Evaluating the effect of a delegated versus centralized control style on the maintainability of object-oriented software," *IEEE Transactions on Software Engineering*, vol. 30, no. 8, pp. 521–534, 2004.
- [33] L. Prechelt, B. Unger-Lamprecht, M. Philippsen, and W. F. Tichy, "Two controlled experiments assessing the usefulness of design pattern documentation in program maintenance," *IEEE Transactions on Software Engineering*, vol. 28, no. 6, pp. 595–606, 2002.
- [34] E. S. Chung, J. I. Hong, L. James, M. K. Prabaker, J. A. Landay, and A. L. Liu, "Development and evaluation of emerging design patterns for ubiquitous computing," in *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods and Techniques*, ACM, 2004.
- [35] B. Unger and W. F. Tichy, "Do design patterns improve communication? An experiment with pair design," in *Proceedings of the International Workshop on Empirical Studies of Software Maintenance*, 2000.
- [36] P. G. Polson, E. Muncher, and G. Engelbeck, "A test of common elements theory of transfer," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM Press, Boston, Mass, USA.
- [37] M. P. Robillard, W. Coelho, and G. C. Murphy, "How effective developers investigate source code: an exploratory study," *IEEE Transactions on Software Engineering*, vol. 30, no. 12, pp. 889–903, 2004.
- [38] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard et al., "Preliminary guidelines for empirical research in software engineering," *IEEE Transactions on Software Engineering*, vol. 28, no. 8, pp. 721–734, 2002.
- [39] D. M. Berry and W. F. Tichy, "Comments on formal methods application: an empirical tale of software development," *IEEE Transactions on Software Engineering*, vol. 29, no. 6, pp. 567–571, 2003.
- [40] C. B. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 557–572, 1999.
- [41] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern Oriented Software Architecture*, John Wiley, Chichester, UK, 1996.
- [42] P. Puschner and A. Burns, "Writing temporally predictable code," in *Proceedings of the 7th International Workshop on Object-Oriented Real-time Dependable Systems*, pp. 85–91, 2002.
- [43] P. Puschner, "The single-path approach towards WCET-analysable software," in *Proceedings of the IEEE International Conference on Industrial Technology (ICIT '03)*, pp. 699–704, December 2003.
- [44] J. O. Coplien, *Software Patterns*, Copyright 1996 AT&T, Copyright 2000, Lucent Technologies, Bell Labs Innovations, 2000.
- [45] A. Albert and R. Bosch, "Comparison of event-triggered and time-triggered concepts with regard to distributed control systems," in *Proceedings of the Embedded World Conference*, pp. 235–252, Nuremberg, Germany, 2004.
- [46] P. Puschner, "Is WCET analysis a non-problem? Towards new software and hardware architectures," in *Proceedings of the 2nd International Workshop on Worst Case Execution Time Analysis*, Vienna, Austria, 2002.
- [47] A. K. Gendy and M. J. Pont, "Towards a generic, "Single Path Programming" solution with reduced power consumption," in *Proceedings of the International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE '07)*, pp. 65–71, Las Vegas, Nev, USA.
- [48] M. J. Pont, *Embedded C*, Addison-Wesley, 2002.
- [49] A. K. Gendy and M. J. Pont, "Automating the process of selecting an appropriate scheduling algorithm and configuring the scheduler implementation for time-triggered embedded systems," in *Computer Safety, Reliability and Security*, M. Harrison, Ed., Lecture Notes in Computer Science, Springer, Newcastle Upon Tyne, UK, 2008.
- [50] A. K. Gendy and M. J. Pont, "Automatically configuring time-triggered schedulers for use with resource-constrained, single-processor embedded systems," *IEEE Transactions on Industrial Informatics*, vol. 4, no. 1, pp. 37–46, 2008.