Empirically Evaluating the Impact of Applying Aspect State Machines on Modeling Quality and Effort

Shaukat Ali[1,2], Tao, Yue[1], Lionel Briand[1,2]
[1]Certus Software V&V Center, Simula Research Laboratory,
P.O.Box 134, 1325, Lysaker, Norway
[2]Department of Informatics, University of Oslo, Norway
{shaukat, tao, briand}@simula.no

***Abstract*—Aspect-Oriented Modeling (AOM) has been the subject of intense research over the last decade and aims to provide numerous benefits to modeling, such as enhanced modularization, easier evolution, higher applicability as well as reduced modeling effort. However, these benefits can only be obtained at the cost of learning and applying new modeling approaches. Studying their applicability is therefore important to assess whether they are worth using in practice. In this paper, we report the first controlled experiment to assess the applicability of AOM, focusing on a recently published UML profile (AspectSM). This profile was originally designed to support model-based robustness testing in an industrial context but is applicable to the behavioral modeling of other crosscutting concerns. This experiment assesses the applicability of AspectSM from two aspects: the quality of derived state machines and the effort required to build them. With AspectSM, a crosscutting behavior is modeled using so-called "aspect state machine". The applicability of aspect state machines is evaluated by comparing them with standard UML state machines that directly model the entire system behavior, including crosscutting concerns. The quality of both aspect and standard UML state machines derived by subjects is measured by comparing them against their corresponding reference state machines. Results show that aspect state machines derived with AspectSM are significantly more complete and correct though AspectSM took significantly more time than the standard approach, probably due to a lack of familiarity of the subjects.**

Keywords: Aspect-Oriented Modeling; Controlled Experiment; Applicability; Robustness; UML State machines

I. INTRODUCTION

Aspect-Oriented Modeling (AOM) aims to provide enhanced Separation of Concerns (SoCs) during design modeling [1, 2]. Crosscutting concerns, for example related to robustness or security behavior, are modeled as aspect models and are subsequently woven into their primary/base model capturing non-crosscutting concerns, such as nominal functional behavior. AOM is expected to yield benefits such as improved readability, enhanced modularization, easier evolution, improved model quality, increased reusability of models, as well as reduced modeling effort [2]. However, there to date is very little evidence supporting such benefits. Empirical investigations, such as controlled experiments and case studies, are required to support the above claims and better comprehend AOM's limitations. This paper, as part of a larger set of studies, is a first step in this direction and reports of the first controlled experiment assessing of the applicability of AOM.

In industrial models such as state machines, one must not only model nominal behavior but also robustness behavior, for example describing how the system should react to abnormal environmental conditions. This is for example needed—and that was our original motivation—to support the model-based robustness testing of embedded or communication systems [3], though there are many more possible applications. In a previous paper, we reported on AspectSM [4], a UML profile for AOM, which was defined to model crosscutting behaviors using *extended* UML state machines, with the objectives of minimizing modeling effort and the learning curve for modeling crosscutting behavior. The AspectSM profile focuses on UML state machines as they are the extensively used notations in practice, for example in model-based test case generation in the context of control and communication systems [3, 5]. Comparable approaches [6-9] in the literature do not use UML extension mechanisms and make use of specific notations for aspect-related features that do not follow any standard. With our industrial partners, and generally in most realistic settings, it is necessary to provide AOM support based on the UML standard to facilitate adoption. A detailed comparison of the AspectSM profile with other related profiles can be found in [4]. AspectSM has been successfully applied to model robustness behavior of video conferencing systems for the purpose of model-based robustness testing at Cisco Systems Inc., Norway [5].

Crosscutting behavior, such as robustness behavior in industrial systems, can result in cluttered UML state machines. As a result, modeling such crosscutting behavior directly on UML state machines can be error-prone and entails significant modeling effort due in part of extensive redundant modeling. Consistent with AOM claims, using AspectSM to model crosscutting behavior as aspect state machines separately from the base state machine should reduce cluttering in models and hence improves the overall quality of the models, which thereby increases the applicability of the approach. Studying the applicability of AspectSM is therefore essential to demonstrate that its use is beneficial to ease its adoption in practice. In addition, from a more general standpoint, studying the applicability of AspectSM provides preliminary evidence about the benefits that can be obtained with AOM. We therefore performed and report here on the first controlled experiment to evaluate the applicability of AspectSM by comparing it with directly modeling crosscutting behavior using standard UML state machines. The controlled experiment was conducted with 25 graduate students taking a graduate course in 'Advanced Software Architecture' at the University Institute of Information Technology (UIIT) at the Pir Mehr Ali Shah Arid Agriculture University, Rawalpindi, Pakistan. The original design (specification and state machines) of the case study system we used in the experiment was defined independently from our experiment—a Automatic Teller Machine (ATM) control system provided in a well-known textbook [10] — but we had to model the case study with three additional crosscutting behaviors: cancel transaction, network failure and power failure. The quality of state machines is measured with three objective measures: *Completeness*, *Correctness* and *Redundancy*, which are proposed for this experiment but can be reused to perform similar experiments in the future, either with AspectSM or other state machine-based approaches. Furthermore, we evaluate the effort (measured in time) that subjects spent to model crosscutting behaviors using standard UML state machines and aspect state machines. Experiment results show that modeling crosscutting behavior as aspect state machines significantly increases completeness and correctness of models and significantly reduces redundancy as compared with standard UML state machines. On the other hand, aspect-oriented modeling took significantly more time than standard UML state machines.

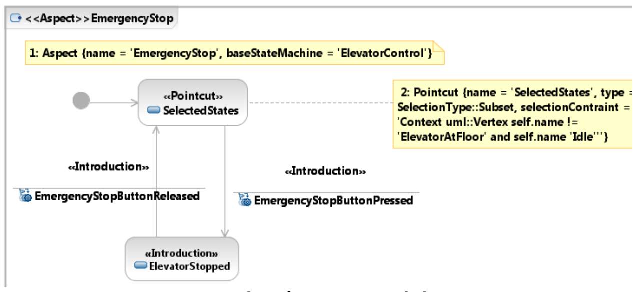
The rest of the paper is organized as follows. Section II describes the necessary background on aspect state machines. Section III provides details on the experiment planning and Section IV reports on results and discussions. Section V provides possible threats to validity of our experiment and Section VI compares related controlled experiments in Aspect-oriented Software Development (AOSD) to our experiment. Finally, we conclude our paper in Section VII.

II. Modeling Aspect State Machines

AspectSM [11] is a UML profile, which was proposed to support the modeling of system robustness behavior—a very common type of crosscutting behavior in many types of systems such as communication and control systems [2]. An example of a robustness behavior for a communication system is related to how the system should react, in various states, in the presence of high packet loss. The system should be able to recover lost packets and continue to behave normally in a degraded mode. In the worst case, the system should go back to the most recent state and not simply crash or show inappropriate behavior. In a control system, one needs to model, for example, how the system should react, in various states, when a sensor breaks down. AspectSM allows modeling crosscutting behavior as aspect UML state machines. Such an approach, relying on a standard and using the target notation (i.e., UML state machine in our context) as the basis to model the aspects themselves, is expected to make the practical adoption of aspect modeling easier in industrial contexts. In our previous work [11], we thoroughly compared AspectSM with similar existing AOM profiles and we observed that AspectSM is the only approach that is exclusively based on standard UML notation, thus eliminating the need for learning additional non-standard notations or languages, and therefore making it easier to reuse open source and commercial modeling tools. This is highly important in most industrial contexts and strongly affects the adoption of modeling technologies. In addition, it is easier to train engineers in standard languages such as UML as many of them have been exposed to it during their university education and previous work experiences.

Though AspectSM was originally defined to support scalable, model-based, robustness testing, including test case and oracle generation, a fundamental question is whether it is easier to model crosscutting concerns such as robustness with AspectSM than simply relying on UML state machines to do it all. In AspectSM, the core functionality of a system is modeled as one or more standard UML state machines (called base state machines). Crosscutting behavior of the system (e.g., robustness behavior) is modeled as aspect state machines using the AspectSM profile. State machines developed using this profile will be referred to as aspect state machines. A weaver [11] then automatically weaves aspect state machines into base state machines to obtain a complete model, that can for example be used for testing purposes. The AspectSM profile specifies stereotypes for all features of AOM, in which the concepts of *Aspect*, *Joinpoint*, *Pointcut*, *Advice*, and *Introduction* [2] are the most important ones, which are specified as stereotypes of the AspectSM profile. Interested readers may consult [4], where further details of the profile are provided.

Below, we present an example of the application of AspectSM. An aspect state machine modeling crosscutting behavior *EmergencyStop* is shown in Figure 1. This UML state machine is stereotyped as <<*Aspect*>>, which means that it is an aspect state machine. The <<*Aspect*>> stereotype has two attributes: *name* and *baseStateMachine*,



<<Aspect>>EmergencyStop
1: Aspect {name = 'EmergencyStop', baseStateMachine = 'ElevatorControl'}
«Pointcut»
SelectedStates
2: Pointcut {name = 'SelectedStates', type =
SelectionType::Subset, selectionContraint =
'Context uml::Vertex self.name !=
'ElevatorAtFloor' and self.name 'Idle'''}
«Introduction»
EmergencyStopButtonReleased
«Introduction»
EmergencyStopButtonPressed
«Introduction»
ElevatorStopped

Figure 1. An aspect state machine for crosscutting behavior EmergencyStop

Table I. Complexity of the reference state machines modeling the crosscutting behaviors of the case study

Crosscutting behavior	Reference base state machine		Reference standard state machine		Reference aspect state machine		
	# of states	# of transitions	# of states	# of transitions	# of states	# of transitions	# of Pointcuts
Cancel Transaction	6	13	6	15	7	15	2
Network Failure			7	22	8	16	2
Power Failure			7	21	8	15	2

whose values are shown in the note labeled as '1' in Figure 1. The name attribute contains the name of the aspect (*EmergencyStop* in this example), whereas the *baseStateMachine* attribute holds the name of the base state machine, on which this aspect will be woven, which is *ElevatorControl* in this example.

The aspect state machine consists of two states: *SelectedStates* and *ElevatorStopped*. *SelectedStates* is stereotyped as <<*Pointcut*>>, which means that this state selects a subset of states from the base state machine. There are three attributes of <<*Pointcut*>>, whose values are shown in the note labeled as '2' in Figure 1. The *name* attribute indicates the name of the pointcut and type denotes the *type* of the pointcut, which is *Subset* in this case. In AspectSM, different types of pointcuts can defined, a complete list of other types of pointcuts is presented in [11]. The third attribute *selectionConstraint* contains a query in OCL on the UML state machine metamodel, which selects all states of the base state machine except *ElevatorAtFloor* and *Idle*. All the model elements stereotyped as <<*Introduction*>> (one state, two transitions) will be newly introduced elements in the base state machine during weaving. This aspect introduces the *ElevatorStopped* state in the base state machine, and selects all states of the base state machines except *ElevatorAtFloor* and Idle (via *SelectedStates*) and introduces transitions from them to *ElevatorStopped* with trigger *EmergencyStopButtonPressed*. In addition this aspect introduces transitions from *ElevatorStopped* to all the states selected by *SelectedStates* with trigger *EmergencyStopButtonReleased*.

III. Experiment Planning

This section discusses the planning of the experiment according to the definition and reporting template defined by Wohlin et al. [12]. Section III.A provides goals, research questions, and hypotheses; Section III.B provides details on the participants of the experiment, whereas Section III.C provides details on the material we used for the experiment. Section III.D provides metrics that we used to assess the quality of aspect state machines with standard UML state machines. Section III.E discusses the design of the experiment, whereas Section III.F and Section III.G describe the procedure that we followed to conduct the experiment. Last, in Section III.H, we discuss how we select statistical tests for analyzing experiment data.

A. Goal, Research Questions and Hypotheses

The objective of our experiment is to assess the AspectSM profile with respect to its applicability to model crosscutting behaviors as aspect state machines. Applicability is assessed from two aspects: the quality of derived state machines and the effort required to model crosscutting behaviors. We measure the quality of state machines from three complementary points of view: completeness, correctness, and redundancy of an aspect state machine and a standard UML state machine with respect to their corresponding reference state machines.

Based on the objective of our experiment, we defined the following four research questions.

RQ 1: Does the use of AspectSM improve the completeness of state machines with respect to reference state machines, when compared to standard UML state machines?

We wish to compare the completeness of aspect state machines modeling crosscutting behaviors with standard UML state machines modeling the same crosscutting behaviors. The quality of both the aspect and standard state machines is measured against reference state machines as aspect and standard state machines are not directly comparable. None of the expected differences between them can a priori be certain to be in a specific direction. This therefore leads to the definition of a two-tailed null hypothesis:

H^1_0: Completeness of aspect state machines is the same as that of standard UML state machines.

RQ 2: Does the use of AspectSM improve the correctness of state machines with respect to reference state machines, when compared to standard UML state machines?

This research question aims to compare the correctness of aspect state machines and standard UML state machines with their corresponding reference state machines. This leads to the following two-tailed null hypothesis.

H^2_0: Correctness of aspect state machines is the same as that of standard UML state machines.

RQ 3: Does the use of AspectSM reduce the redundancy in state machines with respect to reference state machines, when compared to standard UML state machines?

This research question leads to the following two-tailed null hypothesis.

H^3_0: Redundancy in aspect state machines is the same as that of standard UML state machines.

RQ 4: Does the use of AspectSM reduce the modeling effort?

The previous three research questions looked at measuring the quality of state machines, whereas this research question is concerned with the effort required to model crosscutting behaviors. This leads to the following two-tailed null hypothesis.

H^4_0: The effort to model aspect state machines is the same as that of standard UML state machines.

B. *Participants*

The controlled experiment was conducted at the Pir Mehr Ali Shah Arid Agriculture University, Rawalpindi, Pakistan. The subjects in the experiment were 25 graduate students taking a graduate course in 'Advanced Software Architecture' at the University Institute of Information Technology (UIIT). The course is offered in the Master of Science program. The students in this degree already hold a Bachelor in Computer Science or Information Technology and have already been exposed to the UML notation and extensions in the form of UML profiles. On average, each student went through five development and two modeling courses. Eighteen students (out of twenty-five) have used the UML notation for their final year projects before the experiment was conducted. Twenty students gained development experience in IT companies or as teaching staff in computer science.

Our motivation in selecting this group of subjects was to find participants with adequate background (e.g., UML modeling) that could be trained to use our AOM approach over a short period of time. Our goal was to assess AspectSM with fully trained, competent participants in order to assess the maximum potential benefits of the approach. Most industrial practitioners have very little knowledge of AOP and even less of AOM. Ensuring they have the required background is also difficult. This is why we relied on a group of mature and trained graduate students. The subjects were free to choose to participate or not into the experiments and were told their choice would have no effect on their course grades. All students underwent specific, additional training for the experiments (Section III.0). Five students decided not to participate in the experiment.

C. *Material*

1) Case Study System: The system used for the experiment is the popular automated teller machine (ATM) system reported in [10]. Detailed specifications and the corresponding state machine are presented in Appendix A, but the complexity of the reference state machines for both aspect and standard UML state machines in terms of numbers of states and transitions is shown in Table I. The state machine modeling nominal behavior will be referred to as base state machine in this paper as it is modeling the core behavior of the ATM system. This state machine was provided to the subjects and they were asked to model the following crosscutting behaviors.

Cancel Transaction: A customer may cancel a transaction at any time except when the ATM is closed down or not idle. Whenever a cancel request is made, the transaction is terminated and then the card is ejected.

Network Failure: The ATM's behavior in the presence of network failure is also important. Whenever the network connection fails during the ATM operation, except when it is closed down, the current transaction is saved in its local memory and it then tries to recover the network connection. If the network connection is established, the saved transaction is loaded and the ATM continues the transaction. Otherwise, it will simply remain closed down.

Power Failure: In the case of power failure, then the ATM starts its Uninterruptible Power Supply (UPS) and continues the operation.

2) Answer Sheet: Two answer sheets were developed to collect answers for two groups: one for the group using standard UML state machines (Standard) to directly model crosscutting behaviors on the base state machine and the second for the group modeling crosscutting behaviors using aspect state machines (Aspect). Each answer sheet was designed such that subjects can provide their solution one after another and provide the time required to model each crosscutting behavior.

D. *Dependent Variables*

In this section, we present the dependent variables and their justification.

1) State machine completeness (Completeness): This variable measures the completeness of a subject's state machine by comparing it with a reference state machine. It is determined by the completeness of states and transitions-two main modeling elements of a state machine. Note that, since we have two sets of results with respect to two different treatments: standard state machines and aspect state machines, two sets of measures were designed to evaluate the completeness of two different types of state machines derived by the subjects given different treatments.

The formula for CompletenessT (completeness of transitions) is shown in Table II, which calculates the transition completeness of a subject's state machine by looking at the matched transitions of a subject's solution with the reference solution (this holds for each group). The measures needed for deriving the quality measures in Table II are presented in Table III. Matching of transitions is determined by looking at whether the source and target states of a transition match to the source and target states of any transition in the reference model. Three model elements constituting a transition (i.e., guard, trigger, and effect) are further assessed to evaluate the completeness of a matched transition. Matching of the trigger, guard, and effect of a transition is determined whether their names are the same or similar to the corresponding elements of the matched transition in the reference state machine. The completeness of a matched transition is calculated based on the proportion of the matched trigger, guard, and effect of the transition. For instance, if only the guard and trigger of a transition matches with a transition in the reference solution, then this means that the transition matches 66% (2/3) of the reference transition. For standard state machines, we compare only the guard, trigger, and effect of a transition; while for aspect state machines, we also check whether required stereotypes are applied to transitions. This is so because AspectSM requires applying stereotypes on states and transitions in aspect state machines (Section II). For each matched transition *k* in a subject's solution, we check if its guard, trigger, or effect is missing with respect to the matched transition in the reference solution (Table III). For each missing guard, trigger, and effect, we assign value 1 to the corresponding variable (M_{Tguard_k}, $M_{Ttrigger\_k}$ or $M_{Teffect\_k}$), otherwise 0.

Similarly, we calculate CompletenessS (completeness of states) as shown in Table II. The overall completeness (Completeness) of the state machine is therefore calculated based on the completeness of states and transitions. A simpler way to do so is to simply take the average of CompletenessT and CompletenessS. However, since the numbers of states and transitions in a solution might be different, taking the average might not be appropriate.

Table II. Quality measures for a state machine diagram

Category	Measure	Formula	Formula
Completeness	CompletenessS	$1\text{-}M_S$	$\frac{(N_{t\_r} * \text{CompletenessT} + N_{s\_r} * \text{CompletenessS})}{N_{t\_r} + N_{s\_r}}$
	CompletenessT	$1\text{-}M_T$	
Correctness	CorrectnessS	$\text{CompletenessS}*(1\text{-}I_S)$	$\frac{(N_{t\_r} * \text{CorrectnessT} + N_{s\_r} * \text{CorrectnessS})}{N_{t\_r} + N_{s\_r}}$
	CorrectnessT	$\text{CompletenessT}*(1\text{-}I_T)$	
Redundancy	RedundancyS	N_{extras_s}/N_{s_s}	$\frac{(N_{t\_s} * \text{RedundancyT} + N_{s\_s} * \text{RedundancyS})}{N_{t\_s} + N_{s\_s}}$
	RedundancyT	N_{extrat_s}/N_{t_s}	

Table III. Measures to derive quality measures in Table II

#	Measure	Specification
1	M_{Sname_k}	Missing name of the k_{th} state in a subject's state machine diagram
2	$M_{Sstereotype\_k}$	Missing stereotype of the k_{th} state in a subject 's aspect state machine diagram
3	M_{Tguard_k}	Missing guard of the k_{th} transition in a subject 's state machine diagram
4	$M_{Ttrigger\_k}$	Missing trigger of the k_{th} transition in a subject 's state machine diagram
5	$M_{Teffect\_k}$	Missing effect of the k_{th} transition in a subject's state machine diagram
6	$M_{Tstereotype\_k}$	Missing stereotype of the k_{th} transition in a subject's aspect state machine diagram
7	M_S	For standard state machine: $\frac{\sum_{k=1}^{n}(M_{Sname\_k})}{n}$
		For aspect state machine: $\frac{\sum_{k=1}^{n} (M_{Sname\_k}+M_{Sstereotype\_k})/2}{n}$
8	M_T	For standard state machine: $\frac{\sum_{k=1}^{n} (M_{Tguard\_k}+M_{Ttrigger\_k}+M_{Teffect\_k})/3}{n}$
		For aspect state machine: $\frac{\sum_{k=1}^{n} (M_{Tguard\_k}+M_{Ttrigger\_k}+M_{Teffect\_k} + M_{Tstereotype\_k})/4}{n}$
9	I_{Sname_k}	Incorrect name of the k_{th} state in a subject's state machine diagram
10	$I_{Sstereotype\_k}$	Incorrect stereotype of the k_{th} state in a subject's aspect state machine diagram
11	I_{Tguard_k}	Incorrect guard of the k_{th} transition in a subject's state machine diagram
12	$I_{Ttrigger\_k}$	Incorrect trigger of the k_{th} transition in a subject's state machine diagram
13	$I_{Teffect\_k}$	Incorrect effect of the k_{th} transition in a subject's state machine diagram
14	$I_{Tstereotype\_k}$	Incorrect stereotype of the k_{th} transition in a subject's aspect state machine diagram
15	I_S	For standard UML state machine: $\frac{\sum_{k=1}^{n}(I_{Sname\_k})}{n}$
		For aspect state machine: $\frac{\sum_{k=1}^{n}(I_{Sname_k} + I_{Sstereotype_k})/2}{n}$
16	I_T	For standard state machine: $\frac{\sum_{k=1}^{n} (I_{Tguardk}+I_{Ttrigger\_k}+I_{Teffect\_k})/3}{n}$
		For aspect state machine: $\frac{\sum_{k=1}^{n} (I_{Tguard\_k}+I_{Ttrigger\_k}+I_{Teffect\_k} + I_{Tstereotype\_k})/4}{n}$
17	N_{s_s}	# of states in a subject's state machine diagram
18	N_{t_s}	# of transitions in a subject's state machine diagram
19	N_{extras_s}	# of extra states in a subject's state machine diagram
20	N_{extrat_s}	# of extra transitions in a subject's state machine diagram
21	N_{s_r}	# of states in the reference state machine diagram
22	N_{t_r}	# of transitions in the reference state machine diagram

M_{Sname_k}, $M_{Sstereotype\_k}$, M_{guard_k}, $M_{trigger\_k}$, M_{effect_k}, $M_{Tstereotype\_k}$,I_{name_k}, $I_{Sstereotype\_k}$, I_{guard_k}, $I_{trigger\_k}$, I_{effect_k} , and $I_{Tstereotype\_k}$ are Boolean variables that take value 0 and 1 only. 'n' refers to the number of matched states or transitions.

Considering each modeling element (state or transition) having the same weight, we calculate the overall completeness based on the proportions of states and transitions in a state machines. To achieve this, first we obtain the overall completeness of transitions by multiplying CompletenessT with the total number of the transitions in the reference model (N_{t_r}). Similarly, we calculate overall completeness of states by multiplying CompletenessS with the total number of states N_{s_r}. Finally, we take sum of both and divide it with the sum of the numbers of states and transitions in the reference state machine. Notice that our metrics give equal weights to each type of model elements of state machines (e.g., states and transitions). The metrics for each individual type of model elements are combined together to indicate the overall completeness and correctness of state machines. At this stage of research, it is difficult to precisely devise a different weight pattern.

2) State machine correctness (Correctness): Correctness of a state machine is determined based on correctness of states and transitions. CorrectnessT (i.e., correctness of matched transitions) is calculated by the formula shown in Table II. In this formula, we compare a subject's solution with the reference solution and for each matched transition, we determine if its contained modeling elements are correct. Modeling elements contained by a transition include guard, trigger, and effect. For aspect state machines, additionally, we check whether stereotypes and their attributes applied on each matched transition is correct. For instance, if only the stereotype is incorrect, then the correctness of the transition will be 75% (3/4). Finally, because correctness partly depends on completeness, the overall correctness of transitions (CorrectnessT) is obtained by multiplying CorrectnessT with CompletenessT as we assume missing transitions to be incorrect. For instance, if completeness is 60% for a subject's solution, then we calculate correctness only for the matched 60% transitions and the remaining unmatched transitions (40%) are also considered incorrect. Total correctness (Correctness) is calculated in the similar way as we do for Completeness.

3) Redundancy in state machines (Redundancy): Redundant states and transitions are the ones in a subject's solution that do not match with any model elements in the reference solution. Redundancy is calculated based on redundancy in states (RedundancyS) and transitions (RedundancyT). Redundant states are measured with RedundancyS calculated as N_{extra_s} /N_{s_s} , where N_{extras_s} is the number of extra states identified by a subject and N_{s_s} is the number of states identified by a subject. Similarly, we calculate RedundancyT as shown in Table II and Table III. Finally, the overall redundancy (Redundancy) in state machines is calculated in a similar fashion as Completeness and Correctness.

Since our goal is to devise a set of objective metrics to measure the quality of state machines constructed by subjects, we compare them with reference state machines, so that subjective evaluation can be reduced to a minimum. We therefore make possible the comparison of models derived by different subjects in such a way that identical results would be obtained by different persons measuring a model. In addition, these metrics are not specific to this experiment and the profile under evaluation (AspectSM) and hence they are reusable and can be applied to other experiments that involve measuring the quality of UML state machines.

4) Required modeling effort (Effort): It is the time (minutes) taken by a subject to model each crosscutting behavior. It was simply measured as Completion time – Starting time.

E. Design

The design of our experiment is summarized in Table IV. We used a between-subjects design [12] due to the limited number of tasks we could run within time constraints. During the training sessions (Section III.0), each subject was equally trained to understand the two different types of state machines: aspect state machines (Aspect) and standard state machines (Standard). After the training sessions but before the actual experiment tasks, the subjects were also given an assignment to practice designing state machines. This assignment was marked by the first author of this paper and grades were used to form blocks (i.e., groups of students of equivalent skills). The experiment groups were then formed through randomization and blocking to obtain two comparable groups of 10 students each (*Group 1* and *Group 2*) with similar proportions of students from each block. Each group was provided with the base state machine of ATM (Section III.C) and *Group 1* was asked to model crosscutting behaviors with aspect state machines, whereas *Group 2* was asked to model crosscutting behaviors directly on the base state machine. We decided to provide the base state machine to the subjects instead of asking them to model the behavior of the ATM system from scratch based on the textual requirement specifications due to the following two reasons: 1) AspectSM was specifically designed to model crosscutting behaviors and we were only interested in studying the quality of state machines when modeling crosscutting behaviors, 2) Due to time constraints, modeling a complete system from scratch wasn't practically feasible. Note that in the experiment, we ordered the crosscutting behaviors based on their complexity (Table IV) from simple to complex, to enable the subjects to tackle increasingly more complex crosscutting behaviors.

Table IV. Design for the experiment

Crosscutting behavior	Group 1	Group 2
Cancel Transaction	Aspect	Standard
Network Failure	Aspect	Standard
Power Failure	Aspect	Standard

F. *Training*

Subjects were trained by the first author of this paper. Two three-hour sessions were given on the following topics: 1) Recap of UML state machines since subjects were already familiar with this topic preceding the training, 2) Introduction to the Object Constraint Language (OCL), 3) Introduction to aspect-oriented software development (AOSD), and 4) Aspect-oriented modeling (AOM) using the AspectSM profile. Each topic was accompanied with several examples and interactive class assignments. As previously discussed, the subjects were given a home assignment after the training sessions to practice the three state machine modeling approaches and groups were later formed based on the grades of this assignment.

G. *Data collection*

The solutions were collected from the participants and were marked by the first author of this paper. The data was encoded into a JMP [13] data file to perform the statistical analysis. For the experiment, data integrity was checked using the following rule: for the same subjects and for each step, the starting time should precede the completion time, and the completion time of the current task must precede the starting time of the next task. In addition, to avoid mistakes in marking the solutions, the first two authors double-checked the solutions marked by the other. Moreover, for a sample of randomly selected solutions, the first two authors also checked the consistency of the entries in the JMP file with the marks on the answer sheets and no inconsistencies were detected.

H. *Selection of Statistical Tests*

Using statistical testing, we check whether the differences between modeling approaches are statistically significant to determine if we can reject the null hypotheses stated in Section III.A. To check for significant differences between the two approaches under investigation, we performed the non-parametric Wilcoxon rank-sum test [14]. For all statistical tests reported in this section, we used a significance level of α=0.05. We employed this test because our data set meets all the criteria required by this test, which are: 1) observations in the *Aspect* and *Standard state machine* groups are independent of each other, 2) sample size of both groups are equal, and 3) distributions of dependent variables strongly depart from normality, based on the results of the Shapiro–Wilk W test [14] we performed.

IV. RESULTS AND DISCUSSION

In this section, we present results and discussions for each individual research question (Sections IV.A-IV.D), followed by an overall discussion in Section IV.E. Note that in the next sections, we only provide mean values for the dependent variables. However, detailed descriptive statistics are provided in Appendix B.

A. *Completeness of Aspect State Machines (RQ 1)*

Table V shows average percentages of completeness for the three crosscutting behaviors: *Cancel Transaction*, *Network Failure*, and *Power Failure*, respectively, in terms of completeness of transitions (CmT), completeness of states (CmS), and overall completeness (Cm).

We observed from Table V that overall, for *Cancel Transaction*, AspectSM achieved around 9% higher completeness (from 69% to 78%) than the standard state machines modeling approach. The results of the Wilcoxon rank-sum test for *Cancel Transaction* are also presented in Table V, where none of the results are significant as all p-values for completeness measures for *Cancel Transaction* are above 0.05. To further interpret the non-significant results, we conducted power analysis [14]. The result of Cm yielded an estimated effect size of 0.15 (20% of average) to achieve 80% power (see Table X in Appendix C). The observed effect size is 0.05, which is lower than this estimated effect size (0.15), thus explaining the lack of significance. This suggests that we need to collect more

Table V. Results of the Wilcoxon test for all quality measures*

CB	Measure	Mean (Aspect)	Mean (Standard)	Mean Diff. (Aspect-Standard)	p-value
Cancel Transaction	CmT	0.81	0.69	0.12	0.06
	CmS	0.78	-	-	-
	Cm	0.78	0.69	0.09	0.20
	CrT	0.81	0.69	0.12	0.06
	CrS	0.78	-	-	-
	Cr	0.78	0.69	0.09	0.20
	ReT	0	0.33	-0.33	**0.008**
	ReS	0.06	0.38	-0.32	0.17
	Re	0.02	0.37	-0.35	**0.02**
	Effort	22	8	14	**0.0009**
Network Failure	CmT	0.62	0.47	0.15	0.20
	CmS	0.78	0.88	-0.10	0.11
	Cm	0.69	0.51	0.18	0.16
	CrT	0.62	0.47	0.15	0.20
	CrS	0.75	0.88	-0.13	0.053
	Cr	0.68	0.51	0.17	0.156
	ReT	0.25	0.15	0.10	0.57
	ReS	0.28	0.19	0.09	0.35
	Re	0.27	0.17	0.1	0.48
	Effort	15	9	6	**0.364**
Power Failure	CmT	0.88	0.72	0.16	0.65
	CmS	0.86	0.88	-0.02	0.15
	Cm	0.87	0.74	0.13	0.96
	CrT	0.88	0.72	0.16	0.96
	CrS	0.83	0.88	-0.05	0.65
	Cr	0.84	0.74	0.10	0.08
	ReT	0.03	0.27	-0.24	**0.02**
	ReS	0.09	0.38	-0.29	0.06
	Re	0.06	0.30	-0.24	**0.01**
	Effort	16	7	9	**0.01**
All	CmT	0.77	0.63	0.14	**0.02**
	CmS	0.81	0.88	-0.07	**0.01**
	Cm	0.78	0.64	0.14	**0.03**
	CrT	0.77	0.63	0.14	**0.02**
	CrS	0.78	0.88	-0.10	**0.004**
	Cr	0.77	0.64	0.13	**0.04**
	ReT	0.08	0.25	-0.17	**0.008**
	ReS	0.14	0.31	-0.17	0.166
	Re	0.23	0.39	-0.16	**0.02**
	Effort	18	8	10	**0.0001**

* CB: Crosscutting Behavior, Cm: Completeness, Cr: Correctness, Re: Redundancy, CmT: CompletenessT, CmS: CompletenessS, CrT: CorrectnessT, CrS: CorrectnessS, ReT: RedundancyT, and ReS: RedundancyS.

observations, if we want to draw conclusions with confidence for effect sizes below 20% of the average.

For *Network Failure*, overall, aspect state machines are 69% complete, that is 18% more complete than standard state machines. The results of the Wilcoxon rank-sum test for *Network Failure* are shown in Table V, where once again we didn't observe any significant differences as p-values for completeness measures are above 0.05. For *Power Failure*, overall, aspect state machines are 87% complete, that is around 13% more complete than standard state machines. As for the first two crosscutting behaviors, there were no significant differences observed due to small sample sizes as shown by the results of power analysis which are provided in Appendix C.

When we combined observations from all crosscutting behaviors for completeness measures, aspect state machines obtained an average completeness of 78 %, whereas that of standard state machines is 64% (Table V). Due to larger sample sizes, we now observed significant differences between both groups as shown by bold p-values in Table V. For all three completeness measures (CmT, CmS, and Cm), the p-values are this time below 0.05. Overall, for Cm, the p-value is 0.03 and the mean difference is positive hence leading to the conclusion that aspect state machines are significantly more complete when compared to standard UML state machines for modeling crosscutting behaviors. The most plausible cause for such a difference is that the aspect state machines are less complex in terms of number of modeling elements than the standard state machines when modeling crosscutting behaviors (Table I). Though the complexity brought by pointcuts could be a hindrance to the application of aspect state machines, results suggest it does not seem to be the case.

B. Correctness of Aspect State Machines (RQ 2)

For *Cancel Transaction*, we observe from Table V that the overall correctness (Cr) of aspect state machines is 9% higher than standard UML state machines. However, the results of the Wilcoxon rank-sum test reported in Table V show that the differences are not significant. For *Network Failure*, aspect state machines have 17% more correctness than standard state machines, whereas for *Power Failure*, aspect state machines have 10% more correctness than standard state machines. However, the results of the Wilcoxon rank-sum test in Table V show that the differences between aspect and standard UML state machines for *Network Failure* and *Power Failure* are not significant as p-values are above 0.05 for all correctness measures for these two crosscutting behaviors. As for Completeness, the non-significant results of Correctness for individual crosscutting behaviors are probably due to small sample sizes yielding low statistical power as shown by the results of power analysis provided in Appendix C.

When the observations are combined for all three crosscutting behaviors for correctness, we observe significant differences with p-values for correctness measures that are all below 0.05 (Table V). The p-value for overall correctness is 0.04 and the positive mean difference suggests that the aspect state machines yield higher correctness than standard UML state machines. This result is consistent with the result of completeness and is likely due to the lower complexity of aspect state machines when compared to standard UML state machines for modeling crosscutting behaviors (Table I).

C. Redundancy in Aspect State Machines (RQ 3)

In terms of redundancy, one can observe from Table V that overall for *Cancel Transaction*, redundancy of the aspect state machines is 35% less than the standard state machines. For *Network Failure*, we observe 10% more redundancy in the aspect state machines. For *Power Failure*, the aspect state machines yield 24% less redundancy than standard state machines. For all three crosscutting behaviors together, aspect state machines result in 16% less redundancy. The results of the Wilcoxon rank-sum test (Table V) show significant differences between aspect and standard UML state machines for redundancy. For overall redundancy (Re), the p-value is 0.02 and the negative mean difference implies that aspect state machines have significantly lower redundancy when compared to standard UML state machines.

D. Effort for Modeling Aspect State Machines (RQ 4)

From Table V, we can observe that the subjects took more time for modeling all crosscutting behaviors using AspectSM. For example, the subjects took on average 14 more minutes for *Cancel Transaction*, six more minutes for *Network Failure* and nine more minutes for *Power Failure*. The results of the Wilcoxon rank-sum test on the effort of modeling individual crosscutting behaviors (Table V) show that significant differences were observed for *Cancel Transaction* and *Power Failure*, where the subjects took significantly more time to model aspect state machines, as indicated by p-values below 0.05. However, there were no significant differences observed for *Network Failure*. When the observations were combined from all crosscutting behaviors, we observed that the subjects took significantly more time to model aspect state machines as compared to standard UML state machines for modeling crosscutting behaviors (p-value=0.0001). This could be due to a relative lack of experience in modeling aspect state machines using AspectSM when compared to standard UML state machines.

E. Overall Discussion and Concluding Remarks

Based on the experiment results discussed above, we conclude that overall, the completeness and correctness of the aspect state machines derived by the subjects are significantly better than the standard ones for modeling the same set of crosscutting behaviors. In addition, we also observed that the redundancy of the aspect state machines is significantly less than the standard state machines. This is most likely due to the fact that the aspect state machines are less complex in terms of number of modeling elements than the standard state machines for modeling crosscutting behaviors, as visible in Table I.

Regarding modeling effort, we observed that the subjects took significantly more time to design aspect state machines than the ones who designed standard state machines. This may be explained by the limited experience of the subjects regarding AspectSM. We also collected statistics about the most common mistakes subjects did while modeling aspect state machines and observed that most mistakes were either in applying stereotypes or their associated values for attributes. We found that 27% of transitions modeled by subjects have missing stereotypes, missing attributes of stereotypes, incorrect stereotypes, or incorrect attribute values. Such percentage was observed for states as well. This leads to the conclusion that subjects found it difficult to apply the stereotypes required by the AspectSM profile. This suggests that in the future, we probably need to put more attention on the application of stereotypes during training and thus perhaps expect a reduced modeling time and improved modeling quality when using AspectSM.

In conclusion, even though AspectSM took significantly more time, it resulted into higher quality models: significantly better completeness and correctness, and less redundancy than the standard UML state machine modeling approach. More training with AspectSM is expected to further reduce modeling effort and improve the quality of aspect state machines. Based on the above analysis, we recommend using AspectSM for modeling crosscutting behavior to achieve higher quality models.

V. THREATS TO VALIDITY

Below, we discuss the threats to validity of our controlled experiment based on the concepts discussed in [12]. Conclusion validity threats are concerned with factors that can influence the conclusion that can be drawn from the results of the experiments. As with most controlled experiments in software engineering, our main conclusion validity threat is related to the sample size on which we base our analysis. To deal with this, our experiment design required modeling three crosscutting behaviors to maximize the number of observations within time constraints. The other concern is that the quality of state machines can be interpreted in various ways, depending on one's subjective opinion. However we made an effort to minimize subjective judgments and be objective as much as possible by proposing a set of objective metrics to measure quality of state machines by comparing them with their corresponding reference models. By doing so, subjective perceptions can be reduced to minimum and the comparison of models derived by different subjects becomes possible. Additionally, these metrics are general and are therefore reusable, can be applied to multiple experiments and help prevent bias in the evaluation results.

Internal validity threats exist when the outcome of results are influenced by external factors and are not necessarily due to the application of the treatment being studied. Through our experiment design (between-subjects design), we have tried to minimize the chances of other factors being confounded with our primary independent variable: the use of aspect state machines. We avoided any biased assignment of subjects to groups by using blocking based on assignment marks. The main threat of construct validity is that we were not able to investigate all features of aspect-orientation (such as all types of basic advice) in this experiment due to the nature of our crosscutting behaviors.

Two main threats to external validity are related to our experiment and are typical to controlled experiments in artificial settings: 1) Are the subjects representative of software professionals? 2) Is the experiment material representative of industrial practice, in terms of the size of the artifacts we used? Regarding the former, many practitioners have very little knowledge of AOP or AOM in general, and hence require significant training. This is why we chose a group of experienced graduate students with a suitable educational background (Section III.B). In addition, some studies in [15-17] reported on the performance of trained software engineering students for various tasks when compared with professional developers. These differences were not statistically significant when compared to junior and intermediate developers, thus suggesting that there is no evidence that students trained for the tasks at hand may not be used as subjects in place of professionals. In terms of the second threat small case studies and tasks often tends to minimize the differences among treatments. As we see in Table I, for example, for crosscutting behavior *Network Failure*, the standard UML state machine has seven states and 22 transitions. Such numbers are representative of the state machines of classes and small components. However, because crosscutting concerns are expected to have an even higher impact on large models, we expect the use of AspectSM to be even more beneficial in such cases.

VI. RELATED WORK

Most experimentation in Aspect-Oriented Software Development (AOSD) has been conducted to evaluate Aspect-oriented Programming (AOP) when compared to object-oriented programming in terms of development time, errors in development, and performing maintenance tasks. A controlled experiment [18] was performed in industry settings to measure effort and errors using AOP for applying different maintenance tasks related to tracing crosscutting concerns, i.e., the use of logging to record execution of a program. The results showed that aspect-orientation resulted in reducing both development effort and number of errors.

Another experiment is reported in [19], which compares aspect-orientation (AspectJ) with a more traditional approach (Java) in terms of development time for crosscutting concerns. A similar experiment is reported in [20] focusing on development time to perform debugging and change activities on object-oriented programs using

AspectJ. Both of these experiments revealed mixed results, i.e., aspect-orientation has positive impact on development time only for certain tasks. For instance, AOP seems to be more beneficial when the crosscutting concern is more separable from the core behavior.

An experiment is reported in [21], where two software development processes based on a same aspect modeling approach (i.e., the Theme approach) are compared to determine their impacts on maintenance tasks such as adding new functionality or improving existing functionality. The first process (aspectual process) involves generating AO code in AspectJ from Theme AO models, whereas the second process (hybrid process) involves generating object-oriented code in Java from Theme models. Maintenance tasks are measured based on metrics such as size, coupling, cohesion, and separation of concerns. The results showed that on average the aspectual process took lesser time than the hybrid process.

An exploratory study is reported in [22] to assess if AOP has any impact on software maintenance tasks. Eleven software professionals were asked to perform different maintenance tasks using Java and AspectJ. The results of the experiment revealed that AOP performed slightly better than Object-oriented Programming (OOP), but there were no statistically significant results observed. Another exploratory study is reported in [23] to measure fault-proneness with AOP. Three evolving AOP programs were used and data about different faults made during their development were collected. The experiment revealed two major findings: 1) Most of the faults were due to lack of compatibility between aspect and base code, 2) The presence of faults in AOP features such as Pointcuts, Advice, and inter-type declarations was as likely as for normal programming features. The results turned out to be statistically significant. Another exploratory study is reported in [23], which aims to assess if aspects can help reducing effort on resolving conflicts that can occur during model compositions. To do so, they compared AOM with non-AOM in terms of effort to resolve conflicts and number of conflicts resolved on six releases of a software product line. The results of the study showed that aspects improved modularization and hence helped better localize conflicts, which in turn resulted in reducing the effort involved in resolving conflicts.

Our experiment is different from these experiments from several perspectives. First, our experiment focused on the design of the software development life cycle and AOM. Most of the experiments in the literature have focused on comparing AOP with OOP. We evaluated the “applicability” (in terms of completeness, correctness, and redundancy) of crosscutting behaviors modeled as aspect state machines and compared to standard UML state machines. We further compared the effort of modeling crosscutting behaviors using AspectSM and standard UML state machines. In addition, in our recent controlled experiments [24], we evaluated the readability of AspectSM, which is indirectly measured through defect identification and fixing rates in state machines, and the scores obtained when answering a comprehension questionnaire about the system behavior. Results showed that defect identification and fixing rates are significantly better with AspectSM than with both flat and hierarchical state machines. However, in terms of comprehension scores and inspection effort, no significant difference was observed between any of the approaches.

VII. Conclusion and Future Work

Aspect-oriented Modeling (AOM) has been the subject of intense research in the past decade as it is expected to provide several benefits when modeling complex software systems, including enhanced separation of concerns, improved readability, easier model evolution, increased reusability, and reduced modeling effort. However, there is limited empirical evidence regarding such benefits in the AOM research literature.

In this paper, we reported the first controlled experiment that was conducted to evaluate the “applicability” of an AOM state machine modeling approach. We looked at applicability from two aspects: the quality of derived state machines in terms of completeness, correctness, and redundancy, and modeling effort. The specific AOM approach under evaluation is a UML profile (AspectSM) which was specifically designed to model crosscutting behavior (e.g., robustness behavior) using standard UML 2 state machines with a light weight extension for aspect-oriented features. The AspectSM profile was previously applied to an industrial case study for automated state-based robustness testing in Cisco Systems Inc, Norway and is described in [4]. The applicability of state machines modeling crosscutting behavior using AspectSM (aspect state machines) is compared with that of standard UML 2 state machines.

Results showed that the completeness and correctness of aspect state machines is significantly higher than for standard state machines modeling on the same set of crosscutting behaviors (from 64% to 78% and 64% to 77%, respectively). Furthermore, the redundancy in aspect state machines is significantly less than that for standard UML state machines (from 39% to 23%). The most plausible explanation is that aspect state machines are less complex than standard UML state machines in terms of number of modeling elements. We further observed that aspect state machines took significantly more effort to build than standard UML state machines for modeling crosscutting behaviors. However, given the familiarity of subjects with standard state machines and the fact that AspectSM is a new technology for them, we expect that with more AspectSM training and experience this difference might fade away. Based on the results of the experiment, we recommend using aspect state machines for modeling crosscutting concerns, but only on the condition that modelers receive sufficient training. In the future, we are planning to replicate the experiment to obtain larger sample sizes and increase the analysis statistical power.

References

[1] R. E. Filman, T. Elrad, S. Clarke, and M. Aksit, *Aspect-Oriented Software Development*: Addison-Wesley Professional, 2004.

[2] R. Yedduladoddi, *Aspect Oriented Software Development: An Approach to Composing UML Design Models*: VDM Verlag Dr. Müller, 2009.

[3] R. V. Binder, *Testing object-oriented systems: models, patterns, and tools*: Addison-Wesley Longman Publishing Co., Inc., 1999.

[4] S. Ali, L. C. Briand, and H. Hemmati, "Modeling Robustness Behavior Using Aspect-Oriented Modeling to Support Robustness Testing of Industrial Systems," *Accepted for publication in the Systems and Software Modeling (SOSYM) Journal* 2011.

[5] D. Drusinsky, *Modeling and Verification using UML Statecharts: A Working Guide to Reactive System Design, Runtime Monitoring and Execution-based Model Checking*, 1st ed.: Newnes, 2006.

[6] J. Zhang, T. Cottenier, A. V. D. Berg, and J. Gray, "Aspect Composition in the Motorola Aspect-Oriented Modeling Weaver," *Journal of Object Technology,* vol. 6, 2007.

[7] G. Zhang, M. M. Hölzl, and A. Knapp, "Enhancing UML State Machines with Aspects," in *In Proceedings of the 10th International Conference on Model Driven Engineering Languages and Systems (MoDELS)*, 2007.

[8] G. Zhang, "Towards Aspect-Oriented State Machines," in *2nd Asian Workshop on Aspect-Oriented Software Development (AOASIA'06)* Tokyo, 2006.

[9] D. Xu, W. Xu, and K. Nygard, "A State-Based Approach to Testing Aspect-Oriented Programs," in *17th International Conference on Software Engineering and Knowledge Engineering* Taiwan, 2005.

[10] H. Gomaa, *Designing Concurrent, Distributed, and Real-Time Applications with UML*: Addison-Wesley Professional, 2000.

[11] S. Ali, L. C. Briand, and H. Hemmati, "Modeling Robustness Behavior Using Aspect-Oriented Modeling to Support Robustness Testing of Industrial Systems," Accepted for publication in the Systems and Software Modeling (SOSYM) Journal2011.

[12] C. Wohlin, P. Runeson, and M. Höst, *Experimentation in Software Engineering: An Introduction*: Springer, 1999.

[13] JMP, http://www.jmp.com/, 2010

[14] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*: Chapman and Hall/CRC, 2007.

[15] M. Höst, B. Regnell, and C. Wohlin, "Using Students as Subjects—A Comparative Study of Students and Professionals in Lead-Time Impact Assessment," *Empirical Software Engineering,* vol. 5, pp. pp. 201-214, 2000

[16] E. Arisholm and D. I. K. Sjoberg, "Evaluating the Effect of a Delegated versus Centralized Control Style on the Maintainability of Object-Oriented Software," *IEEE Transactions on Software Engineering,* vol. 30, pp. pp. 521-534, 2004.

[17] R. W. Holt, D. A. Boehm-Davis, and A. C. Shultz, "Mental Representations of Programs for Student and Professional Programmers," in *Empirical Studies of Programmers: Second Workshop*, M. O. Gary, S. Sylvia, and S. Elliot, Eds.: Ablex Publishing Corp., 1987, pp. 33-46.

[18] P. Durr, L. Bergmans, and M. Aksit, "A Controlled Experiment for the Assessment of Aspects - Tracing in an Industrial Context," Enschede: University of Twente, CTIT, 2008.

[19] S. Hanenberg, S. Kleinschmager, and M. Josupeit-Walter, "Does aspect-oriented programming increase the development speed for crosscutting code? An empirical study," in *2009 3rd International Symposium on Empirical Software Engineering and Measurement*: IEEE Computer Society, 2009.

[20] R. J. Walker, E. L. A. Baniassad, and G. C. Murphy, "An initial assessment of aspect-oriented programming," in *21st international conference on Software engineering* Los Angeles, California, United States: ACM, 1999.

[21] K. Farias, A. Garcia, and J. Whittle, "Assessing the impact of aspects on model composition effort," in *Proceedings of the 9th International Conference on Aspect-Oriented Software Development* Rennes and Saint-Malo, France: ACM, 2010.

[22] M. Bartsch and R. Harrison, "An exploratory study of the effect of aspect-oriented programming on maintainability," *Software Quality Control,* vol. 16, pp. 23-44, 2008.

[23] F. Ferrari, R. Burrows, v. Lemos, A. Garcia, E. Figueiredo, N. Cacho, F. Lopes, N. Temudo, L. Silva, S. Soares, A. Rashid, P. Masiero, T. Batista, and J. Maldonado, "An exploratory study of fault-proneness in evolving aspect-oriented programs," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1* Cape Town, South Africa: ACM, 2010.

[24] S. Ali, T. Yue, L. C. Briand, and Z. I. Malik, "Does Aspect-Oriented Modeling Help Improve the Readability of UML State Machines?," *Under consideration for a publication in a Journal,* 2011.

[25] L. Thomas, "Retrospective Power Analysis," *Conservation Biology,* vol. 11, pp. pp. 276-280, 1997.

[26] T. Dyba, V. B. Kampenes, J. E. Hannay, and D. I. K. Sjøberg, "Systematic review: A systematic review of effect size in software engineering experiments," *Information and Software Technology,* vol. 49, pp. pp. 1073-1086, 2007.

APPENDIX A: SPECIFICATION AND BASE STATE MACHINE FOR THE ATM SYSTEM

The complete specification of ATM as adopted from [11] is shown below followed by its state machine in Figure 2.

"A bank has several automated teller machines (ATMs), which are geographically distributed and connected via a wide area network to a central server. Each ATM machine has a card reader, a cash dispenser, a keyboard/display, and a receipt printer. By using the ATM machine, a customer can withdraw cash from either a checking or a saving account, query the balance of an account, or transfer funds from one account to another. A transaction is initiated when a customer inserts an ATM card into the card reader. Encoded on the magnetic strip on the back of the card is recognized, the system validates the ATM card to determine that the expiration date has not passed, that the user-entered PIN matches the PIN maintained by the system, and that the card is not lost or stolen. The customer is allowed three attempts to enter the correct PIN; the card is confiscated if the third attempt fails.

If the PIN is validated satisfactorily, the customer is prompted for a withdrawal, query, or transfer transaction. Before a withdrawal transaction can be approved, the system determines that sufficient funds exist in the requested account, that the maximum daily limit will not be exceeded, and that there are sufficient funds at the local cash dispenser. If the transaction is approved, the requested amount of cash is dispensed; a receipt is printed containing information about the transaction, and the card is ejected. Before a transfer transaction can be approved, the system determines that the customer has at least two accounts and that there are sufficient funds in the account to be debited. For approved query and transfer requests, a receipt is printed and the card is ejected. Customer records, account records, and debit card records are all maintained at the server.

An ATM operator may start up and close down the ATM to replenish the ATM cash dispenser and for routine maintenance. It is assumed that functionality to open and close accounts and to create, update, and delete customer and debit card records is provided by an existing system and is not part of this problem. "

The base state machine for the ATM system is shown in Figure 2.

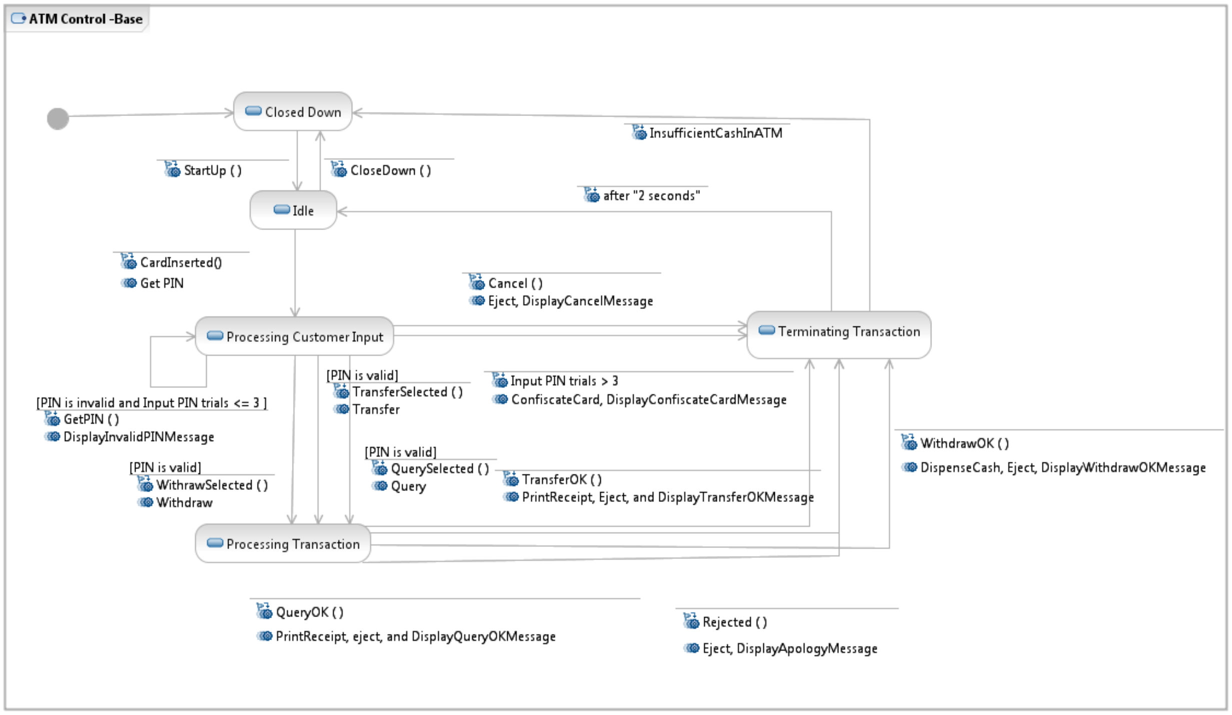


Figure 2. Base state machine for ECS

APPENDIX B: DESCRIPTIVE STATISTICS FOR DEPENDENT VARIABLES

This appendix provides detailed descriptive statistics, i.e., minimum (min), median, maximum (max), mean, and standard deviation (std) for each dependent variable. Table VI, Table VII, and Table VIII provide detailed descriptive statistics for *Cancel Transaction*, *Network Failure*, and *Power Failure* respectively. Table IX provides descriptive statistics for dependent variables, when observations from all crosscutting behaviors are combined together.

Table VI. Descriptive statistics for various measures for *Cancel Transaction*

Descriptive Statistics		**Min**	**Median**	**Max**	**Mean**	**Std**
Measure	**Approach**					
CompletenessT	Aspect	0	1	1	0.8	0.33
	Standard	0.33	0.67	1	0.69	0.19
CompletenessS	Aspect	0.5	0.75	1	0.77	0.2
	Standard	-	-	-	-	-
Completeness	Aspect	0.33	0.83	1	0.79	0.22
	Standard	0.33	0.67	1	0.69	0.19
CorrectnessT	Aspect	0	1	1	0.80	0.33
	Standard	0.33	0.67	1	0.69	0.19
CorrectnessS	Aspect	0.5	0.75	1	0.77	0.20
	Standard	-	-	-	-	-
Correctness	Aspect	0.33	0.83	1	0.79	0.22
	Standard	0.33	0.67	1	0.69	0.19
RedundancyT	Aspect	0	0	0	0	0
	Standard	0	0.5	0.67	0.33	0.28
RedundancyS	Aspect	0	0	0.5	0.05	0.17
	Standard	0	0	1	0.38	0.52
Redundancy	Aspect	0	0	0.17	0.02	0.06
	Standard	0	0.5	0.75	0.37	0.32
Effort (Minutes)	Aspect	15	20	50	23	11
	Standard	5	9	15	8.5	3.5

Table VII. Descriptive statistics for various measures for *Network Failure*

Descriptive Statistics		**Min**	**Median**	**Max**	**Mean**	**Std**
Measure	**Approach**					
CompletenessT	Aspect	0	0.75	1	0.62	0.33
	Standard	0.11	0.47	0.75	0.47	0.20
CompletenessS	Aspect	0.5	0.75	1	0.78	0.2
	Standard	0	1	1	0.88	0.35
Completeness	Aspect	0.2	0.8	1	0.69	0.26
	Standard	0.2	0.5	0.78	0.51	0.18
CorrectnessT	Aspect	0	0.75	1	0.62	0.33
	Standard	0.11	0.47	0.75	0.47	0.20
CorrectnessS	Aspect	0.5	0.75	1	0.75	0.19
	Standard	0	1	1	0.88	0.35
Correctness	Aspect	0.2	0.8	0.9	0.67	0.25
	Standard	0.2	0.5	0.78	0.51	0.18
RedundancyT	Aspect	0	0	1	0.25	0.38
	Standard	0	0	0.75	0.15	0.29
RedundancyS	Aspect	0	0.33	0.75	0.28	0.27
	Standard	0	0	1	0.19	0.37
Redundancy	Aspect	0.2	0.8	0.9	0.67	0.25
	Standard	0.2	0.5	0.78	0.51	0.18
Effort (Minutes)	Aspect	5	16.5	30	9.8	15
	Standard	5	9	15	9.1	3.87

Table VIII. Descriptive statistics for various measures for *Power Failure*

Descriptive Statistics		**Min**	**Median**	**Max**	**Mean**	**Std**
Measure	**Approach**					
CompletenessT	Aspect	0.38	0.88	1	0.88	0.20
	Standard	0	0.88	1	0.72	0.36
CompletenessS	Aspect	0.75	0.75	1	0.86	0.13
	Standard	0	1	1	0.88	0.35
Completeness	Aspect	0.56	0.88	1	0.87	0.13
	Standard	0.11	0.831	1	0.74	0.32
CorrectnessT	Aspect	0.38	0.88	1	0.88	0.2
	Standard	0	0.88	1	0.72	0.36
CorrectnessS	Aspect	0.56	0.75	1	0.81	0.15
	Standard	0	1	1	0.88	0.35
Correctness	Aspect	0.56	0.88	1	0.84	0.13
	Standard	0.11	0.83	1	0.74	0.32
RedundancyT	Aspect	0	0	0.33	0.04	0.11
	Standard	0	0.2	1	0.27	0.32
RedundancyS	Aspect	0	0	0.5	0.09	0.19
	Standard	0	0.5	1	0.37	0.35
Redundancy	Aspect	0	0	0.4	0.06	0.14
	Standard	0.1	0.24	0.9	0.30	0.26
Effort (Minutes)	Aspect	5	10	35	16	10
	Standard	5	5	15	7	4

Table IX. Descriptive statistics for various measures

Descriptive Statistics		**Min**	**Median**	**Max**	**Mean**	**Std**
Measure	**Approach**					
CompletenessT	Aspect	0	0.88	1	0.77	0.3
	Standard	0	0.67	1	0.64	0.28
CompletenessS	Aspect	0.5	0.75	1	0.81	0.18
	Standard	0	1	1	0.88	0.34
Completeness	Aspect	0.2	0.85	1	0.78	0.21
	Standard	0.11	0.67	1	0.64	0.25
CorrectnessT	Aspect	0	0.88	1	0.77	0.3
	Standard	0	0.67	1	0.63	0.28
CorrectnessS	Aspect	0.5	0.75	1	0.78	0.17
	Standard	0	1	1	0.72	0.21
Correctness	Aspect	0.2	0.84	1	0.7	0.21
	Standard	0.11	0.67	1	0.64	0.25
RedundancyT	Aspect	0	0	1	0.09	0.24
	Standard	0	0.16	1	0.25	0.29
RedundancyS	Aspect	0	0	0.75	0.14	0.24
	Standard	0	0	1	0.31	0.41
Redundancy	Aspect	0	0	0.9	0.23	0.34
	Standard	0	0.43	0.9	0.39	0.26
Effort (Minutes)	Aspect	5	19	50	18	11
	Standard	5	7	15	8	4

APPENDIX C: RESULTS OF POWER ANALYSIS FOR NON-SIGNIFICANT RESULTS

Power analysis can be used during the design stage of an experiment to determine how many subjects are likely to be needed, or after the fact to help interpret non-significant results. The latter may be due to small samples sizes and effect sizes that are smaller than expected. Power analysis is particularly important for controlled experiments in software engineering that involve human subjects, as they normally suffer from small sample sizes because of the limited availability of trained subjects and the high cost of conducting experiments. In our context, like in most software engineering experiments, the number of subjects is imposed by external constraints and a retrospective power analysis, as suggested in [25], helps interpret non-significant results in such conditions. For each statistical test considered, such an analysis estimates the minimum effect size at which we can observe an acceptable level of power (typically 80%). This means that above that minimum, we can probably interpret a non-significant result as an absence of effect. Below this threshold the effect might be present but remain undetected.

In our experiment, we are interested in comparing the aspect state machines with standard UML state machines. We perform power analysis for the dependent variables that did not yield significant results and followed the method of calculating power as reported in [25], which requires a fixed sample size, a set significance level (0.05) and power level (80%), and uses the observed variance to calculate the corresponding, minimum effect size. We didn't use standardized effect sizes as suggested by Cohen [26] since those cannot be easily interpreted in a software engineering context.

Table X summarizes the results of power analysis. The table shows the estimated effects size thresholds corresponding to 80% power for measures that yielded non-significant results (*Minimum effect size*). This means that for effect sizes less than these thresholds, power is less than 80% thus entailing a significant risk of error (type II) in not rejecting the null hypotheses. In other words, for effect sizes below these thresholds, we cannot draw conclusions with confidence from the statistical test results presented in Table V. The *Average* column in Table X shows the average values for the dependent variables, when combining all the observations being compared from both approaches. The last column shows the percentage of *Average* that corresponds to the minimum effect size. For example, in Table X, the result of Completeness for Network Failure yielded an estimated effect size of 0.17 (28% of average) to achieve 80% power. The observed effect size is 0.09, which is lower than this estimated effect size (0.17), thus explaining the lack of significance. This suggests that we need to collect more observations, if we want to draw conclusions with confidence for effect sizes below 28% of the average. Similar results are observed for other measures for individual crosscutting behaviors, which are summarized in Table X.

Table X. Results of power analysis

Crosscutting Behavior	Measure	p-value	Observed Effect Size	Minimum Effect Size	Average	Minimum Effect Size/Average
Cancel Transaction	Completeness	0.20	0.05	0.15	0.74	0.20
	Correctness	0.20	0.05	0.15	0.74	0.20
Network Failure	Completeness	0.16	0.09	0.17	0.6	0.28
	Correctness	0.156	0.08	0.16	0.59	0.27
	Redundancy	0.48	0.08	0.16	0.59	0.27
Power Failure	Completeness	0.96	0.07	0.18	0.81	0.22
	Correctness	0.08	0.05	0.18	0.79	0.23