

# Display Techniques for Multiple Feature-Sets in State Charts

Rorik Henrikson  
University of Toronto  
rorik@dgp.toronto.edu

## ABSTRACT

With different versions of a product having similar UML diagrams, it is generally easier and more efficient to combine these similar charts into one UML diagram to depict all of the differences and similarities in the product models. In this paper we introduce a set of principles for creating these multiple feature-set state charts; to design standardized charts that are easy to read and easy to understand. We discuss the different aspects we explored, and the user study we performed to further assess the recommendations.

## 1. INTRODUCTION

In Product design, it is common to see different product lines from a company, each version of a product having its own, features, each product being slightly different from the others. These features define how this product is unique from similar products. This feature-set helps designers and engineers create the needed components for a specific product model. These features can affect many different job areas – software engineering included.

In software engineering (SE), the unique features of a product need to be presented in different UML diagrams. The problem is that there is no standardization for how to display this individual information. Consequently everyone demonstrates unique features in their own way. Since features can be almost anything (i.e. triggers, actions, properties, states, timings, etc.), any UML diagram could potentially have the need to demonstrate distinct features.

Since each UML diagram has its own specifications and rules, each UML diagram would need to be assessed individually. In this work we have focused on defining techniques to demonstrate the distinct differences between feature models in a combined state chart (see figure 1). We have tried to create a set of principles that a software engineer can follow when creating a combined state chart that will help make the different features in a specific feature-set obvious, easy to identify, and quick to understand.

In this paper, in section 2, we start by defining what we mean by a feature-set. We continue in section 3 by exploring existing situations where multiple levels of information need to be presented to a reader, and existing attempts at this type of graph have been performed. We continue in Section 4 by identifying key priorities and constraints in a graph that will effect user perception. We also examine the different components of a state chart, how we might present the different components and how these aspects can be combined to form a full state chart with the combined feature-sets. In section 5 we discuss a user survey we performed testing these different ideas, then wind-down in section 6 and 7, discussing further research opportunities and

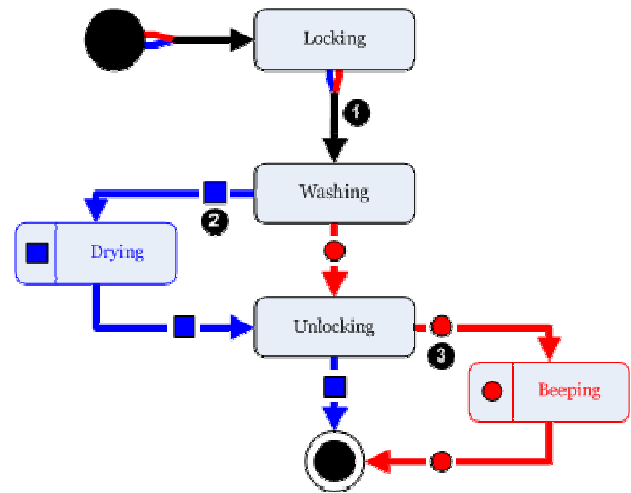


Figure 1. A simplified multiple feature-set state chart for two washing machine models using index markers

observations we made during our study. Finally, we conclude in section 8 summarizing our suggestions and findings.

## 2. MULTIPLE FEATURE-SET

Each product that is created by a company has its own unique factors – these include things such as abilities, features, properties, design, timings, etc. A list of all these factors, including the factors that deviate from other products creates a “feature-set” that identifies a particular product.

Each feature-set would consequently have its own set of UML diagrams. If one were to, combine all the UML diagram of one type for a single product line (with all the different versions) one would end up with a diagram that demonstrates multiple feature-sets all at the same time. This diagram would potentially be more informative and easier to read than having to read one-by-one all of the individual diagrams for each version of a product in a product line.

Since each type of UML diagram has its own “rules” and “properties”, each UML diagram type would need to be addressed individually. For this work we have focused on state-charts. However, the principles of this work should be easily transferable to other UML diagram types.

## 3. RELATED WORK

One main area of inspiration where designers have needed to compress large amounts of information into layers on a single document is with maps. Both subway and road maps have been facing these problems for decades, and have been able to try many different ideas – we have tried to draw inspiration from some of their lessons. For example the Tokyo, London and Paris subway

map have complex interactions, as do many of the European highway and city maps.

We have also examined work where other authors have drawn up models to try and display multiple feature-set depictions. In particular we used models from Classen, et. al [1], Heidenreich, Sánchez, et. al [4] and Rubin and Chechik [5] to compare against our approaches and which we presented in the user study. We also analyzed approaches by Kästner et. al. [2], Heidenreich and Wende, [3] and Czarnecki and Antkiewicz [6] for further ideas.

## 4. STATE CHARTS

Since each type of UML diagram has unique rules and properties, each UML diagram needs to be addressed individually to examine their properties and to determine the best way to combine elements to show specific features related to the given feature-set for the specified model. In this work, we have examined state charts, the connectors between states, the states themselves, and the information associated with the components of a diagram. We also examine some general principles of creating full state charts.

### 4.1 Priorities and Considerations

In looking at methods for displaying information in a compressed format, there were a few priorities that we wanted to maintain. The first goal was to have the information being portrayed by these diagrams as obvious and intuitive as possible. Secondly, the different parts of the diagram, which feature-set is being displayed, what components belong to which feature-set, and the states associated with a given set should all be easy to identify. Third, the overall result should be that these graphs are quick and easy to understand.

We also wanted to create a system that would be scalable and be approached similarly for a diagram with two feature-set models, or one with 30 feature-set models. The obvious constraint when combining 30 feature-set models is that there is a significant amount of information to relay to the reader. To maintain readability, this means that the graphs need to be kept as free from clutter as possible. This also creates a challenge to ensure that any information and the markers used to indicate this information are kept in an appropriate proximity to the object to which they refer.

### 4.2 Connectors

Examining maps, there are many different ways to label different paths. Colour is definitely a key factor, however, cannot be used as a sole indicator due to colour blindness and grey-scale printing. Also roads and subway lines usually have names or numbers associated with them. To identify this information on a map, there are many different methods used to attach numbers and possibly symbols to the appropriate path.

For UML state charts, numbers are not overly useful for labeling connectors due to the fact that numbers have other meaning,

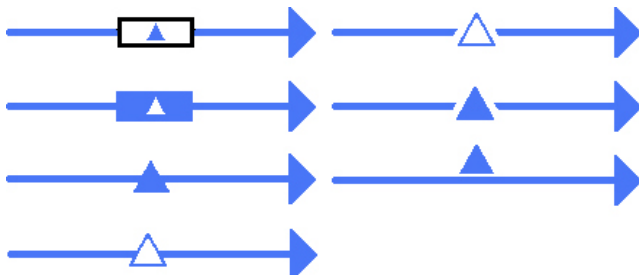


Figure 2. Possible single connector arrows

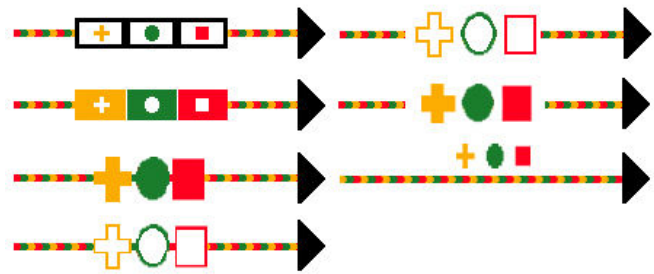


Figure 3. Possible multi connector arrows

however, symbols (such as circles, squares, crosses, triangles, etc.) can easily be used to associate a specific model with a path.

These paths can represent a connection between two states that is individual to one feature-set, belonging to several feature-sets, or part of all feature-sets.

#### 4.2.1 Single Connector Paths

Single connector paths are the easiest to show as they can be represented by a single colour and a single symbol. The question becomes, what is the best way to associate and display this symbol on the path? We have explored seven different methods of displaying this information (see figure 2), from which we queried the opinion of the users in our user study to help select the best representation. The results of the survey are further discussed in section 5.

#### 4.2.2 Multiple Connector Paths

Multiple connector paths are used when a subset of several features-sets connect two states. In this case, it is important to differentiate between different paths. For consistency, one should include the appropriate symbols associated with the features-sets in the same method that one shows the symbol for a single feature-set connector. However, in the case of multiple features-sets, one would need to show all the symbols for the current path. One of the decisions with multiple connector paths is choosing the colour for the arrow. Again, it is important for a state chart to be scalable; therefore a solution that works for two objects must also work for 30 objects. The obvious solution is simply drawing several arrows one above the other to create a striped band with the appropriate colours. This quickly becomes unmanageable after about 5 models have been combined in one connector – but even after 3 models, the graph starts to get cluttered and unclear. Upon examination of this problem, it quickly became apparent that it is possible to stripe the arrow in the colours for that path (see figure 3). This provides two visual cues. First, the line being striped indicates that this is not a single item, but rather there are multiple arrows “intertwined” on this path. Second, it allows the appropriate colours to be associated with the connector. Even if this is printed in grey scale, the stripping is an indicator to look for the symbols. Like with the single connectors, we asked in our user study which arrow people preferred, so as to help find the optimal representation. This is discussed further in section 5.

#### 4.2.3 All Paths

There is a special case scenario with these types of connectors – the case when all feature-sets share the same path. In this case, adding the symbols and all the colours to a connector is unnecessary, and simply adds visual noise and clutter. In this



Figure 4. Combined arrow used for all paths

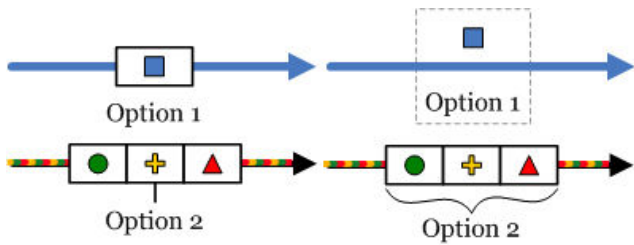


Figure 5. A subset of possible labeled arrows

situation, it is possible to combine the paths into a single black arrow (see figure 4). For this arrow, on the one side (coming from the initial state), we see many paths starting out and joining together mimicking what might be seen as the “tail” of an arrow. These paths then combine into a single line forming an arrow which continues the rest of the distance to the second state. These initial colour paths serve two purposes. First, if you’re visually following a colour, it more intuitively leads your eye along the path, as the colour still exists to get you started. Secondly, regardless of colour, it makes the arrow look distinct from the single and multiple case arrows. This means that the reader does not need to spend time processing what they are seeing. They know that all the paths are covered by this case.

#### 4.2.4 Labeling Paths

Knowing that a given path is associated with a specific feature-set model is useful and necessary; however, often one needs to provide information with the specified arrow. This means that labels of some sort need to be applied to a given arrow. We explored several different methods to associate text with the given arrow (a small subset of these options is shown in figure 5). Interestingly, our user study seemed to show that displaying text underneath the symbol without any extra lines to create an association is sufficient while reading a diagram.

### 4.3 States

In state charts with multiple feature-sets, many states will likely be shared. There, however, are often states that are unique to a given feature-set, or shared by only a few feature-sets. At other times a state may be missing from one model, but present for another. This “missing node” condition is a special case that we discuss further below.

#### 4.3.1 Combining Conditions

It is common for one product to have a similar progression of states to another model with a middle state that is slightly different. For example, product A may go from state 1 to state 2 to state 4. Product B may go from state 1 to state 3 to state 4. In this case, both models are ultimately going from state 1 to 4; however, one model uses state 2 as an intermediary step while the other model uses state 3 as the intermediary. The obvious way to show this on a state chart is for each model to have its own path creating a pattern that is similar to that of a diamond with each state being at a vertex (state 1 and 4 at either end, and state 2 and 3 occupying the top and bottom vertex). This gets harder to manage when there are many feature-sets being displayed each with a unique middle state, and introduces a significant amount of clutter to a chart.

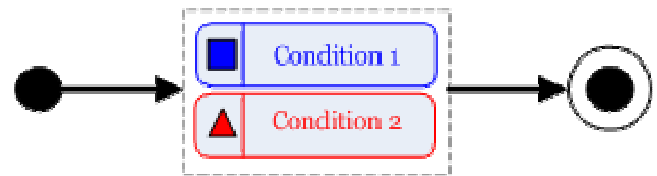


Figure 6. One possible method for combining two states into one representation

We looked to see if it was possible to combine these states into one graphical representation with the information pertaining to each individual model being displayed (see figure 6). We have tried several different approaches some showing promise in our user study. However, it is apparent that more reviews need to be done to form any conclusive statement.

#### 4.3.2 “Missing” Conditions

Similar to the above situation, it is possible that a feature-set will have a state that does not exist in other feature-set models. This means that, due to necessity, the default way is to have two separate paths. We wanted to see if it was possible to combine the paths and still indicate this one “missing” state from the other models in an intuitive manner.

To do this, we enter the middle/missing state as if it applied to all the feature models following the specified path. Rather than having information in this state for every node, we only indicate the node for the model that actually does have a state at this location. To leave the state, we use the same arrow we used to enter. (see figure 7)

Our user survey indicated that this isn’t as intuitive as we hoped and could potentially cause confusion. Further testing, however, needs to be done to see if this is the case after a user has learned the specified technique.

#### 4.3.3 Identifying Nodes

In the same way that we identify the connectors between states, it is important to be able to identify the states themselves and associate them with specific feature-sets.

To do this, we approach it in a similar way to how we handled the connectors. First, the text and bounding shape should be coloured the colour associated with that specific feature-set. If there is a mix of several colours, one can simply use black. Second, we show the symbol for the particular feature-set on the left hand side of the state box (see the “Barking” state in figure 7). In this way, we associate both the colour and the symbol of the feature-set to the state.

### 4.4 Full Charts

Looking at components individually is useful for evaluation; however, something that seems like a good idea in isolation may not work well in context. Therefore, we took several state charts that had been developed by other authors and reworked them using the principles that we established in the above steps. This gave us a good indication of what worked well and what needed to be further explored. Much of our summarizing was confirmed by our user study (see section 5 for more details).

Further, there are a few principles that need to be applied to an overall state chart that are not apparent when working with the individual components.



Figure 7. graph indicating a “missing” state



	
1	Trigger: filled to wtrLevel; heated to 30C;
2	Action: /dryer.sendSignal(SigStart)
	
1	Trigger: filled to wtrLevel; heated to 30C;
3	Action: /dryer.sendSignal(SigStart)

Figure 8. An example table with index markers

#### 4.4.1 Displaying Extra Information

When combining only a couple of feature-set models, it is usually easy to fit the information needed on a graph; however, once and a while there is more information than fits in the space available. This also is an issue if you are combining many feature-set models as, even if each model only has one line of information, this can quickly fill the vacant space.

To accommodate for this, we introduced a simple technique of marking a location with a smaller black circle (or “index marker” – see figure 1) with a number in it that is sequential to the order that it appears in the progression of the state-chart. One can then create a table off to the side with a section for each feature-set. Each section can have the feature-set symbol in the header of the section, with the index markers along the side. The table can then be populated with the information that is to be shown at the specified location (See Figure 8). Yet, this is not without cost. Since putting the extra information in a chart means that you are likely putting the information off to the side, this means that the user must make an extra step to identify the needed information, where to find it, and then visually leave their current spot and re-locate it after the fact. This is something that many people do not like doing.

#### 4.4.2 Most Likely Path

When building a state chart, one should try to ensure that the states that are most common to all feature-sets are the easiest to follow and pick-out. As a result, the most likely path should contain this information that is most common. A feature-set that has a unique entity should display this entity off of the main branch of the state chart.

If there is a tie in how popular a path is (i.e. two feature-sets use one path, two others use a different path), it would be logical to split the difference and have one path go up, and the other go down forcing the user to pay attention to which path they are following.

#### 4.4.3 Cardinal Directions

When designing state charts, it is always difficult to know where to place components on the chart. However, for simplifying readability, the eye is naturally drawn along straight lines. Therefore, for easy of reading, it would be logical to have the path that is easiest to follow be the path that is “most common” to all features-sets.

To accomplish this, since most state charts occupy more than one line, one should try to stick to the primary cardinal direction, and then use the next most common angles (i.e. 45°, 135°, etc.) when

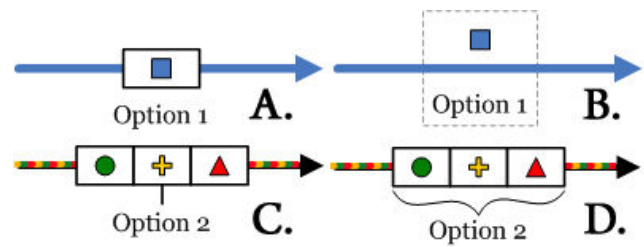


Figure 9. Several methods for associating information with connectors

more flexibility is needed. The more “right-angled” the chart, the easier it will be to read and “pick-out” specific features.

## 5. USER SURVEY

To evaluate our techniques we created an online survey to get feedback from users. In total we had 9 participants representing a group of graduate student in Computer Science at a large North American university. The survey was divided into 3 parts - a part dealing with connectors, a part to deal with states, and a part to examine state charts as a whole. We created two versions of the survey and on the second survey presented the information, in each part, in a reversed order to help accommodate for a biasing effect. Unfortunately only 2 participants answered the second questionnaire. Finally the survey used a within participant model. Our goal with this initial survey was to get a sense of which techniques users preferred and which were intuitive.

For evaluating connectors, we presented a few questions representing the different versions of single feature-set connectors (see Figure 2), several questions representing the multiple feature-set connectors (see Figure 3), and some questions representing both connectors together. Each type was presented both in isolation and in context. In each case, we asked the users for their preference based on which representation they found clearest.

The most surprising result from this section was that people generally did not like the technique where the symbol for a connector was above the line (see Figure 2, 3<sup>rd</sup> down on right) when viewed in isolation. For the in-context questions, we did show a smaller subset of the available options, however, the “above” version was one of the more popular techniques for this scenario. In general, the preferred method for both multiple and single feature-set connectors is the solid shape embedded in the connector arrow with a bit of space to either side of it (see Figure 2, second down on right). The next most preferred method was tied – both the connector with a solid box on the line – a shape embedded in reverse within it (see Figure 2, 2<sup>nd</sup> down on left) and the “above” technique were equally popular.

The next section of the survey asked about the “combined arrow”, used when all models use the same path (see section 4.2.3). Encouragingly, most people seemed to intuitively understand this arrow, and follow the key features of this representation. One person even correctly identified “four flows of different products merge” – which is essentially what the arrow is showing.

We followed this by examining different techniques for associating information with the connectors. For this question, we only presented a small subset of the possibilities due to the sheer volume of the number of different cases we would need to present for all possibilities. As well, these questions were asked out of context. Based on the discrepancy between in-context and out-of-context from the initial questions in the survey, this should

be further examined in a follow-up survey. This question had the user rate the proposed methods from 1 to 4 based on their preference for the demonstrated technique. The most popular technique for associating this information with a single connector is by surrounding the label and symbol with a dotted line (see Figure 9B), followed by having no extra symbols – simply having the text underneath. For multiple feature-set arrows, the most popular method was to use a brace (see Figure 9D), closely followed by tie for multiple connector lines, and no extra markings. The fact that participants seemed to be comfortable with no extra markings associating the information below a connector to the connector itself is promising as extra marks will add clutter to a diagram.

Combining states met with mixed reaction, and we feel that this area still has room for more improvement. We had users rate 5 different techniques on a scale of 1 – 5 based on how clear they felt the representation was. The section showed mix results with the multiple states embedded in a dashed-line being the most popular (see Figure 6). This question, however, only explored the case combining two states. A follow-up survey should also pursue larger scenarios to get a proper representation of the technique.

Based on the reactions and guesses that people had regarding the “missing state” scenario (see Figure 7), showed that this technique is not necessarily intuitive. The main concern would be that some people thought that this representation might mean that the state is optional versus essential for the specified feature-set. We did not explain the meaning in the survey as part of the goal of the survey was to discover what is intuitive. This representation may not have issues once it has been explained. This would need to be explored in a follow-up survey.

Finally we took combined feature-set models created by other authors [1][4][5] and re-created them using some of the techniques explored (see Appendix A). We presented these models to the participants and had them rank the different models and comment on their likes and dislikes of the different forms. Ultimately these graphs need to be followed-up in a second survey as, based on some of the comments, it is possible that preferences from previous questions affected users views of the combined charts. For example, if someone did not like the dotted-line box around the symbol and supporting text, they were more likely to dislike a figure containing this symbolism (see figure G in in Appendix A). We had one user point out that they found this image to be “a bit noisy”. Ultimately, these graphs should have used the techniques that were preferred in part 1 and 2 of the survey to avoid biasing. It should also be noted that not all graphs for comparisons shared all the “same” basic features. For example, one graph, we had the details written on the original, but used the index markers for the redrawn graph (see figures D and E in appendix A). One user did not like having to look away for the information and commented that they preferred the other due to the fact it did not “make me look away in terror [of losing my spot]”

Finally it should be noted that due to time constrains we ultimately tried to cover too many different aspects in too short of a survey. We also, unfortunately, did not have enough participants to show any significant difference between the different results, but the respondents gave us good general feedback on what they were understanding, what was confusing, and where more work needs to be done. As well, we did not explore the black and white aspect of these images, which influenced some of the design decisions. As has been noted a couple of times, a follow-up

survey should be performed to try and tease out remaining aspects of the new implementation.

## 6. FUTURE WORK

The work in this paper represents a good first start. It has established techniques that seem to work well for single, multiple, and all connectors.

Some of the initial exploration for combining states has shown promise, however, due to the mixed reactions to these techniques in our user study, a follow up study needs to be performed. For this study we looked at how people reacted to mixed states without explaining the “rules” or purpose of what was being introduced. This was done on purpose to try and get a sense of what was intuitive and what was confusing. Yet, many of the issues people had would probably be easily solved by a simple explanation of the ideas behind the design. This would need to be confirmed with a follow-up survey.

This study also only looked at small state charts, and though the techniques were designed for large charts as well as small, only small charts were tested in this work. It would be important to run a study with both small and large examples to get a full sense of whether these combined features have the overall desired effect, and are as simple as the initial research seemed to indicate.

Finally part of this design focused on the fact that colour is not necessary. At this point, the black and white representation of these charts has not been tested. This is a factor that should be brought into consideration for fully evaluating the effectiveness of these graphs.

## 7. OBSERVATIONS

While working with the different charts, different suggestions, and people’s comments, it became quickly apparent that the intended use for the UML model makes a difference on how people read and interpreted the diagrams. If the reader’s goal is to read the information quickly, they are likely more interested in a model that has as little clutter as possible with as few distractions as possible. They would likely want information quickly, and not have to waste more time searching for features than necessary. However, if a reader’s goal is to quickly identify the differences in one model or another, they may not care about the extra details and just want to be able to take in the different paths as quickly as possible. If a user simply wants to get an overall feel for how the models relate to one another, they may just be interested in the differences, in which case they would end up looking for deviations from a standard path. If the state chart is going to be used for reference while programming, the reader will likely want as much information as possible at whatever spot they happen to be looking. In this case it is probably beneficial to have as much information crammed into the model as possible.

Whatever the reader’s preference, it is apparent that there are differences in how an individual reads a state chart and what their goal is for the graph while using it. To accommodate the different needs, compromises would need to be chosen that would benefit as many of the different uses as possible.

As well, with our questionnaire, examining the results of the combined principles did not render as much information as might have been hoped. Many of the full charts needed to be redrawn to demonstrate the user’s preferences from the first part of the survey. As it was, many of the problems with the combined questions seemed to reflect the respondents opinion of the



individual parts queried at the beginning of the survey. This could be either due to the fact that these representations do not work well, or it could be a biased based on the layout of the survey. A follow-up survey would help differentiate this difference.

## 8. CONCLUSION

In this initial research we have shown promising techniques for combining multiple feature-sets, the connectors, and the states that belong to them in a combined multiple feature-set state chart. We have shown that a symbol associated with a colour works well for identifying a specific feature-set and recognized that this will help regardless of if the chart is shown in black and white or colour. We have isolated a few methods for displaying single as well as multiple feature-set connectors. We have shown that multiple features-set arrows can be a striped coloured version that is otherwise similar to the single feature-set case. We have introduced a “combining” arrow that simplifies a model even further by representing a situation where all feature-sets share a similar path, which helps reduce clutter.

We have introduced the concept of combining unique states in a state-chart, and combining situations where a state chart has an extra state not present in another feature-set. We have explored several different ways of presenting this information; however, have concluded that more research needs to be done to help clarify the best way to approaching this problem.

Finally we introduced some fundamental rules for laying out state charts to help the user quickly understand what they are observing. We introduced the concepts of keeping the paths to cardinal directions when possible, keeping the most likely path as straight as possible, and introduced “index markers” when there is too much information to display in the available space.

Through a user study, we have explored the new principles introduced, and obtained feedback on what people find easy to follow, what they find confusing, and what needs further study.

Though further work still needs to be done, the work presented demonstrates a promising start in standardizing and presenting multiple feature-set UML diagram state charts in a compressed and easy to understand method.

## 9. ACKNOWLEDGMENT

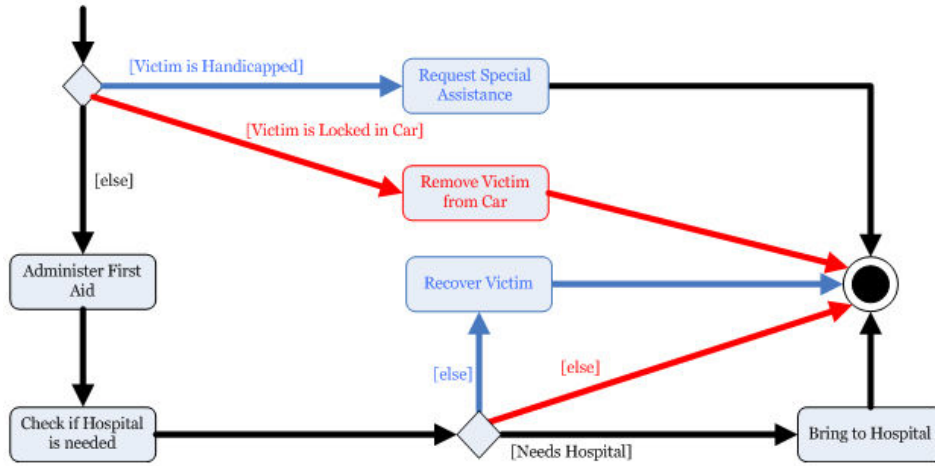
I'd like to thank Julia Rubin for the initial idea and motivation behind this work, and for supplying me with references and comments to further understand this problem.

## 10. REFERENCES

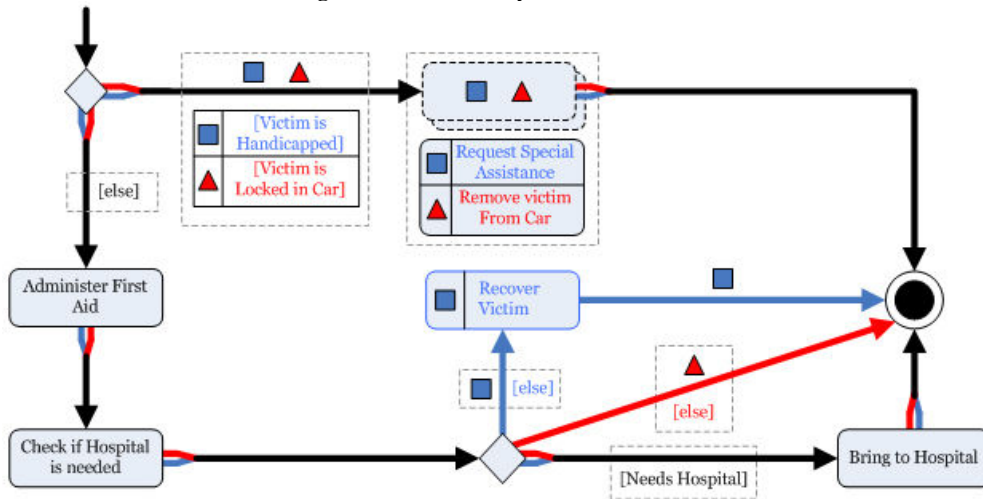
- [1] Andreas Classen, Patrick Heymans, Pierre-Yves Schobbens, Axel Legay, Jean-François Raskin: Model checking lots of systems: efficient verification of temporal properties in software product lines. In *Proceedings of ICSE'10: 335-344, 2010*
- [2] Christian Kästner, Salvador Trujillo, and Sven Apel. Visualizing Software Product Line Variabilities in Source Code. In *Proc. SPLC Workshop on Visualization in Software Product Line Engineering. 2008.*
- [3] Florian Heidenreich and Christian Wende. Bridging the Gap Between Features and Models. In *Proceedings HeidenreichWende-AOPLE07. 2007.*
- [4] Florian Heidenreich, Pablo Sánchez, João Santos, Steffen Zschaler, Mauricio Alférez, João Araújo, Lidia Fuentes, Uirá Kulesza, Ana Moreira, and Awais Rashid. 2010. Relating feature models to other models of a software product line: a comparative study of featuremapper and VML. In *Transactions on aspect-oriented software development VII*, Shmuel Katz and Mira Mezini (Eds.). Springer-Verlag, Berlin, Heidelberg 69-114.
- [5] Julia Rubin, Marsha Chechik. Quality of Merge-Refactorings for Product Lines. *Unpublished.*
- [6] Krzysztof Czarnecki and Michał Antkiewicz. 2005. Mapping features to models: a template approach based on superimposed variants. In *Proceedings of the 4th international conference on Generative Programming and Component Engineering (GPCE'05)*, Robert Glück and Michael Lowry (Eds.). Springer-Verlag, Berlin, Heidelberg, 422-437. DOI=10.1007/11561347\_28 [http://dx.doi.org/10.1007/11561347\\_28](http://dx.doi.org/10.1007/11561347_28).

# Appendix A

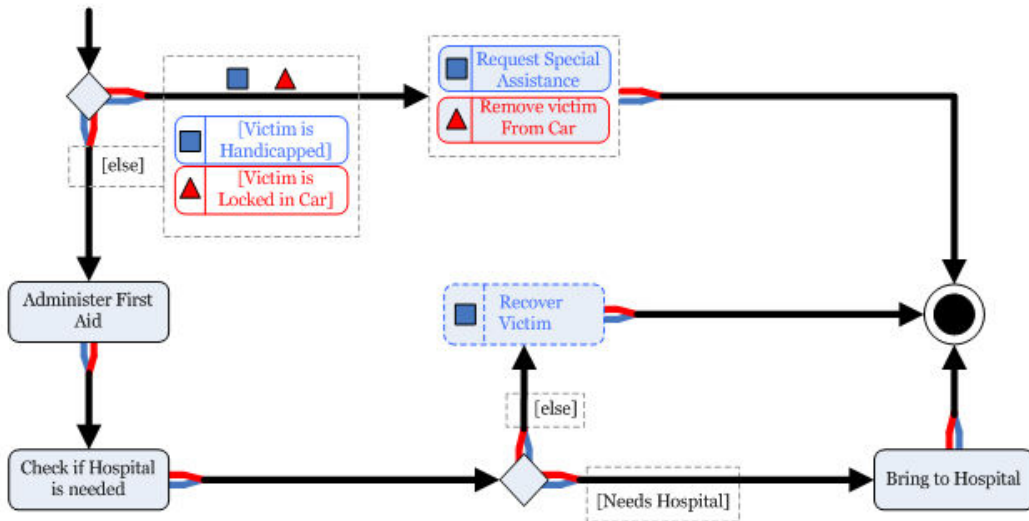
## UML State Chart – Partial Rescue Plan - Model A



A. Original Rescue Plan by Heidenreich et. al.

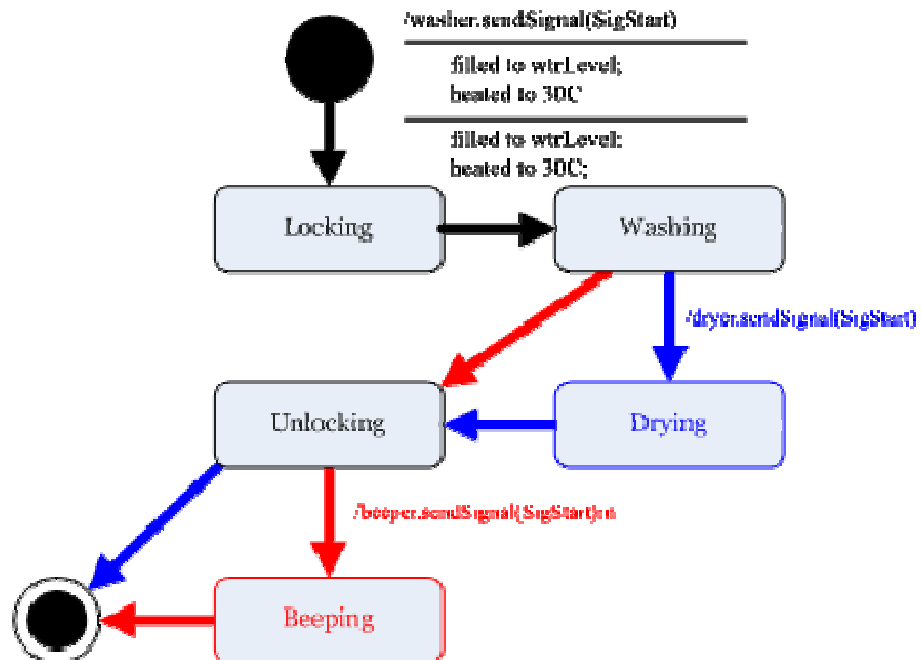


B. One version of the reformatted rescue plan

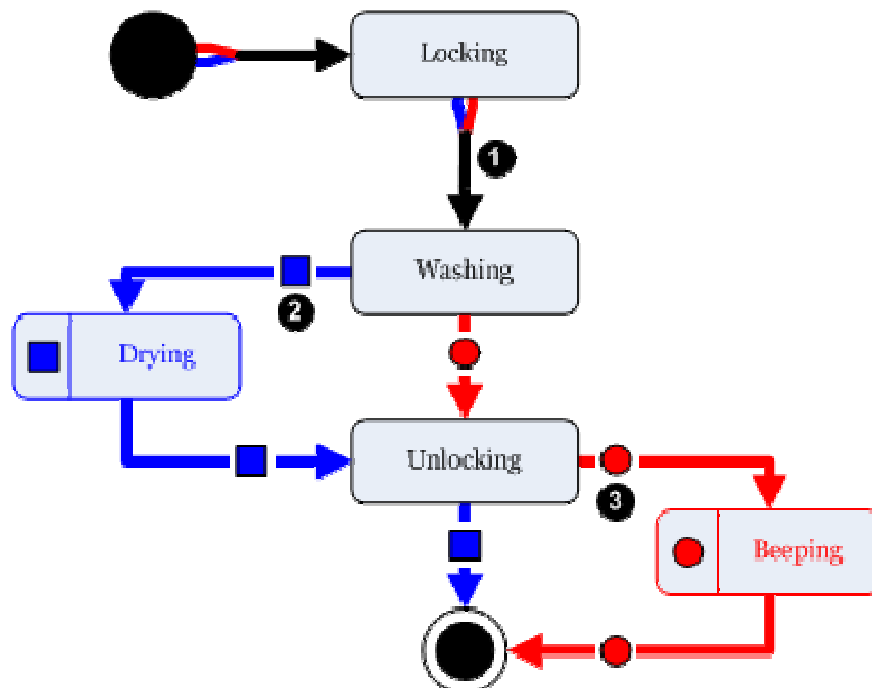


C. A second version of the reformatted rescue plan

## UML State Chart – Washing Machine - Model B



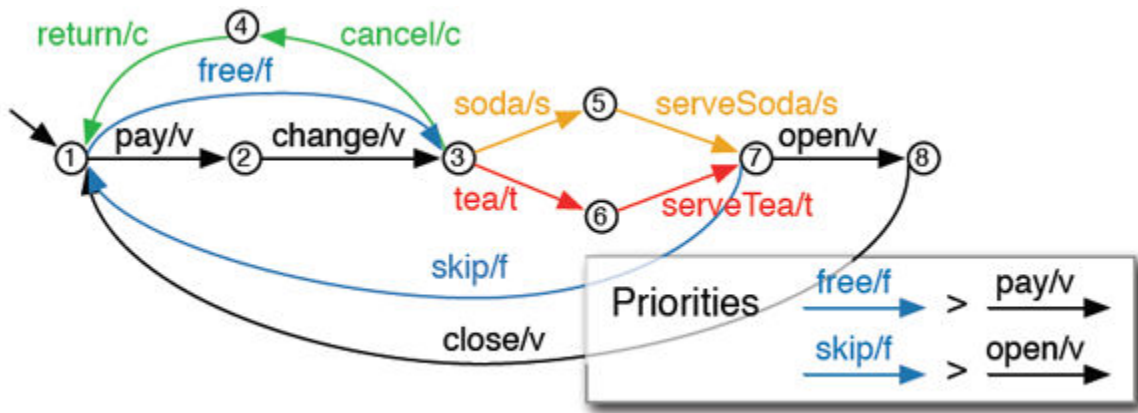
D. The original washing machine model by Rubin and Chechik



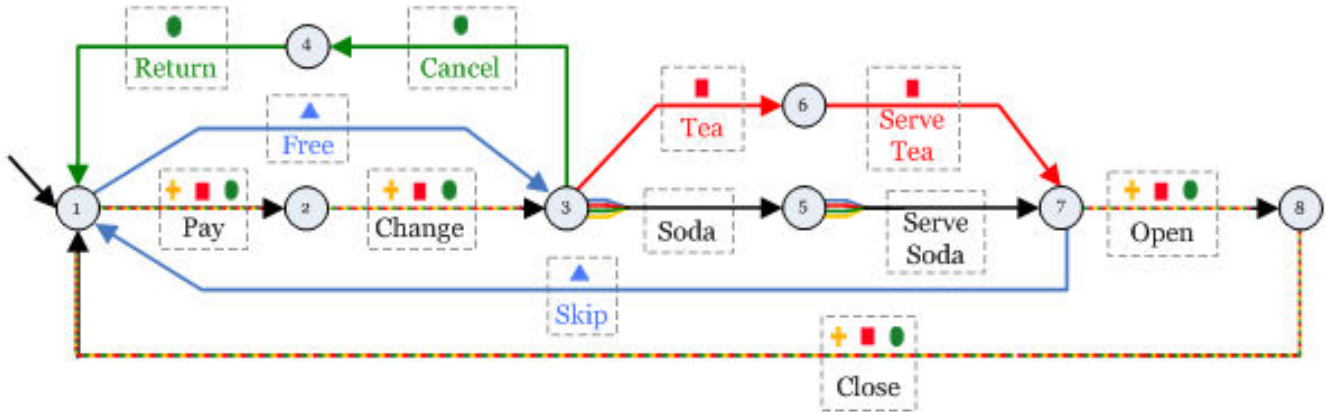
E. The reformatted washing machine model



# UML State Chart – Vending Machine – Model C



F. The original vending machine model by Classen, et. al.



G. The reformatted version of the vending machine