|  |  |
| --- | --- |
|  |  |

**MODELS 2011 Foundations Track Paper Notification [171]**

|  |  |
| --- | --- |
| **Jon Whittle <whittle@comp.lancs.ac.uk>** | **2011年7月6日 上午6:39** |
| 回复: whittle@comp.lancs.ac.uk  收件人: yili07@sei.pku.edu.cn, zhangw@sei.pku.edu.cn, zhhy@sei.pku.edu.cn, zhijin@sei.pku.edu.cn, meih@pku.edu.cn  抄送: whittle@comp.lancs.ac.uk, models2011-papers-webadmin@borbala.com | |
| |  | | --- | | \*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=  First reviewer's review:           >>> Summary of the submission <<<  The paper proposes an approach for merging feature models using semantic information about feature dependencies. The paper presents the merging algorithm and reports results of its evaluation.           >>> Evaluation <<<  This is an easy to read, mostly well-written paper, on a relevant topic. I could understand the problem being tackled, but showing concrete evidence of that problem with a motivating example would be important. Overall, I like the proposed solution and the idea of using semantic information to drive the merging algorithm, but I was disappointed by spotting a few technical imprecisions and not being able to understand the rationale for the approach used to merge constraints. Moreover, I was a little puzzled by the fact that the initial motivation was to merge feature models so that one can deal with large feature models in a modular way (by separately specifying smaller feature models and then using the proposed algorithm to merge them and obtain the desired feature model), but the evaluation seems to focus on merging different versions of similar features models.  For the modularity motivation, I would like to see a discussion on how the approach deals with dependencies between features in different modules, features with the same name denoting different features in different modules (this appears much latter, in Sec 4.8, but should be briefly mentioned earlier on), and the same feature appearing with different parents in different modules. This could be elaborated when describing the overall requirements, in Sec 3.2.  The subset constraint at the beginning of Section 3.3 does not define the semantics of the merging operation, but some property it must satisfy. Many operations might satisfy that, so how do I know which one this paper is proposing? The algorithm defines it, but the semantics of the operation should define it too, at a higher level. This is mostly a terminology issue, but should at least be discussed. More important, Table 1 shows that the expected result in Fig 2 would not strictly satisfy the subset constraint; that is, the target in Fig 2 is not a proper merge following the 'semantics' of the merging operation. In fact, {Screen, High Resolution, Touch} is not a valid configuration of the target FM. High Resolution is not even a feature of the target FM.  It is important to explicitly mention earlier that 'the root feature of both sources must be matched'. What are the implications of this decision? It seems to me that it significantly restrains the use of the merging operation to support modular development of feature models.  Lines 11-13 in the algorithm do not seem to match what is illustrated in Fig 5, which shows how to merge children of different semantic dependencies. Is the name get\_children\_of\_the\_same\_refinement\_semantics really appropriate?  In page 7, I understand that  'Another property is that the rules ensure the configuration set of target FM is exactly the cross-product of source FMs' configuration sets, so the semantics of merging operation is preserved'  is in the right direction. However, whereas that holds for rule 5(a), it does not hold for 5 (b) because new intermediate features were introduced and they appear in the target configurations. The formal definitions should then be changed.  In page 9, I understand  'Table 3 gives definition of the rules, and reasoning of the rules is also based on the cross-product semantics of merging operation'  but this doesn't mean much. Different rules could also be based on the 'semantics'. For example, dropping the constraint of the target model on Rule 9 wouldn't still conform to the semantics? So why to opt for the original Rule 9? It would be important to briefly justify this.  The evaluation section needs a threats to validity subsection.  Even when changing the text, I could not really understand the two points below  The semantics of merging operation is defined by the relation between the configuration set of source and target FMs -> The semantics of the merging operation is defined by a relation between the configuration set of source and target FMs' could not understand  First we represent a configuration of an FM as a set features contained in the configuration. -> First we represent a configuration of an FM as a set of features contained in the configuration. Definition is circular  Here are some more specific comments.  pg 5: semantics information to refinements -> semantic information to refinements  pg 6: feature of target FM is created -> feature of the target FM is created  pg 10: (FS: Feature Set) better to explain in the text, not in the table  Here are my co-reviewer comments:  The paper is well written and in an interesting scope.  The problem could be better motivated.'  In the algorithm section, as you did with the semantics of the merging operation, I would prefer to see an example first, rather than the algorithm, since you introduce the new refinement types in the meantime.'  The example given in section 4.6 is not clear. Isn't it the general rule for cross-product semantics? I failed to notice anything particular about this example, regarding constraints.'  The motivation for merging FMs is given for large FMs, but you only evaluate smaller FMs. Is this related to the manual steps needed in your algorithm?'  Regarding the evaluation, I would also like to see the effort in time for performing the merging operations. This would be good to show if the manual steps aren't a threat for the algorithm.'  It would be good to make the feature model artifacts available online.'  In the first paragraph: "codes" ==> code.'  In the first paragraph of Section 4.2 there is an error when referencing a work.'  It is not clear to me the difference between Decomposition and Specialization, maybe a more precise definition for establishing the criteria for choosing between these alternatives?  For the preprocessing part, the user has to go through all features of the FM? In Other issues (4.8) you mention that renaming imposes considerable work load. Doesn't preprocessing imposes such load too?'  pg 1: the authors point out that, an FM -> the authors point out that an FM  \*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*  Second reviewer's review:           >>> Summary of the submission <<<  The paper presents a technique for merging feature models. The basic property of the merge result is that is represents the cross-product of the valid configurations of the source models. The technique relies on manual annotation of the source models to provide additional feature relationship semantics that is relevant to the merge. Among others, a specialization refinement is turned into a characterization "attribute" if any other kind of refinement is added. The result may require manual post-processing to name newly inserted features and to restructure parts that resulted from hierarchy incompatibilities.           >>> Evaluation <<<  The paper addresses a relevant problem in feature modeling. Model merging is of particular importance to building models, as larger models are often built from smaller models; also extending existing models with new products (special case of a model with just one configuration) can be handled by such merging.  The proposed technique is extremely simple. This is actually its main strength. It also looks promising as to be practical; however, the evaluation is the weakest aspect of this work.  I like the idea of enriching the hierarchies with refinement information. This reminds me of the idea that feature models are "views on ontologies" [see reference below]. In a sense, the annotations reveal some of the underlying ontology, which helps making more informed merge decisions. <http://gsd.uwaterloo.ca/node/76>  It was not entirely clear to me how much one gains by these annotations, though. Thus, I would suggest a better discussion of the benefits and tradeoffs, as the annotations represent manual effort. In particular, I would like to see some example that compare the results of merging with and without the annotations and a discussion of what the annotations helped to do.  As a specific point, I wonder how would you handle the case that where, say in Fig. 5 (b), the second source model would add some resolutions. This would require merging two unnamed characteristics, or better, just merging the two sets of resolutions.  The evaluation is the weakest part of the paper. I like that the approach was tried on a number of non-trivial models that were independently developed. But I was missing more detailed discussion of the experimental setup. Also, I would like to see a more in-depth discussion of qualitative aspects of the results. What were the cases that were handled well and what were the cases that caused problems. I would like to learn from the evaluation the potential limitations and ideas for improvement. Also, the technique assumes manual preprocessing to add domain semantics to model prior merging, but does not evaluate the effort required to do this. The experiment should also discuss the threats to validity of the results.  The current technique description could be compressed to make space for the explanation of the role of the refinements and the improved evaluation section.  The technique has some limitations; in particular: 1) Features are matched purely on name. The authors don't address matching of  features themselves (e.g. synonyms or very similar features). 2) It handles only requires and excludes cross-tree constraints - does not  address other forms of constraints.  Both points could be handled same way as in the related work on reverse engineering feature models (<http://gsd.uwaterloo.ca/node/322>).  Presentation.  - Figures should appear at the top or bottom of a page. - Missing reference to previous work on different semantics of refinements. - Algorithm 1. What happens when merge returns null?  Language and Typos.  - Section 1. The acronym FM is used for both "feature model" and "feature  modeling". Write "feature modeling" out in full when referring to it. - Section 1, page 2. "acceptable features" - what does this mean? - Section 3.3. "First we represent a configuration..." - Change to "First,  we represent a configuration of an FM as a set of features". - Section 3.3. "Then Let..." - Change to "Then, let..." - Section 4.2. "Error! Reference source not found." - Section 6.2. Acher et al. - not Archer et al.  This is what my co-reviewer had to say:  The paper proposes a new operator for merging feature models, or more precisely: it introduces a syntactic extension of feature models, and then offers merging of these exteded models. The nature of the extension is exploited to enable merging, which reflects the semantics of features. On one hand this is an interesting idea, which sheds light on challenges in merging feature models (lack of semantic precision), on the other hand it does not really solve the problem of merging models in a modeling lagnuage as known from industrial tools.  The "semantic" merge operator is called a cross-product, which is a rather misleading name, given that the operator has nothing in common with a cross (cartesian) product. The operator is actually defined by union, not by a cross product, and it seems to be some kind of join. Moreover, and very importantly, the operator is used to state the semantic correctness requirement for the syntactic merge operator.  Unfortunately, this correctness requirement is very weak, which positions the entire construction in the later part of the paper an as an ad hoc construction rather than systemtatically designed method. Note that, the merge operator that would simply create a 'tautology-model" (root, and a union of all other features as optional children of root) satisfies this requirement - it admits all feature configurations of the merged models (and much more!). I would expect a requirement that the operator is as closed to the joined semantics of the source models as possible (some kind of semantic minimality), or an evaluation that proves that it is not far from semantic minimality indeed.  [p. 4] "The semantics defined above ensures the configurations of the target FM are combinations of valid configurations of both sources."  The above stament is incorrect. Take model M1 to have features root, a, b and model M2 to have featues root, a, c. All features a, b, c are optional children of root in their respective models. So {root, a, b} is a configuration of M1, and {root, c} is a configuration of M2. Now {root, a, b, c} is a configuration of M1 ⊗ M2, but it violates exclusion between b and c. In M1 we have a constraint: a requires b; In M2 we have a constriant: b excludes c. Having analyzed the algorithm later, I seem to agree that this problem does not appear, but still your interpretation of your requirements is flawed.  "The algorithm contains two manual steps and two automated steps"  There is no such thing as manual steps in an algorithm (algorithms is a procedure for computation!). You probably talking about procedure, or a method, or a workflow, not an algorithm.  The difference between characterization and decomposition is a bit weak. In class modeling this would essentiallly be just a difference of a type (whether we compose into complex types, or simple types), and in this sense a bit minor. The difference between characterization/decomposition and inheritance (specialization) is of a much stronger nature. So it is not really clear for me whether there is need to distinguish characterization and decompositon.  In fact having analyzed the rules in Fig. 5, I can see that there is no difference between decomposition and characterization - the rules would make sense, and there would be fewer of them if we simply combined decomposition with characterization (essentially creating two refinements only "specializing refinement" and "non-specializing refinement".  One issue is that your algorithm requires the same hierarchical structure of the models. I think the treatment of hierarchy is the main problem in merging feature models, so you should not side-step it. In section 4.8 there is a small comment on this topic: saying that this algorithm does not require parent compatibility, since if the models are not parent compatible, it simply clones parts of the subtree. This is presented as an advantage over algorithms that require parent compatibility. Ufnortunately, this is inconsistent with authors own statement a few lines below, when they admit that parent incompatibility is left out unresolved by the algorithm. Cloning parent incompatible features creates an incorrect model with respect to your semantics. If you assume that a name identifies a feature, then you should not create two features that have the same name. It is very easy to create an input for your algorithm so that the result of mergin will be a model which admits a configuration which both requires and forbids a feature (the cloned feature). I do think however that this issue can be toned down, by putting more emphasis on the entire workflow and not claiming that the algorithm solves this problem, but it enables the user to solve it.  The rules in Table 2 seem strangely non monotonic. You combine mandatory(a) and optional(a) into a mandatory(a) - so strenghtening (or conjoining).  I suppose that this has to do with your ⊗ operator, and its correctness condition. i.e since a will be present in any instance of the second model, then it is safe to assume that it needs to be present in the outcome of merging. However, at the same time, you merge optional and "or" into optional, so you weaken (!) the model. Why cannot you basically maintain the optional ? (the stronger constraint?). This is not explained. More importantly, I believe that if your correctness requirement for merging was defined more strictly, you would discover that this latter design decision is too weak, and inconsistent with a proper requirement.  I think there is one essential problem with the evaluation method: it checks how much information in the merge is superflous and should be removed (characteristic features, clones, constraints). But it does not check how much information in the merge is missing (lost constraints, etc). How the merged models compare to models that would be created manually to represent the same problems? Also, the evaluation is not against the baseline of the existing algorithms, so we cannot really see whether the current algorithm is an improvement. What are the differences with other merging algorithms for FMs ?  PRESENTATION  I do not think that the presentation in the paper in adequate. Section 4.3 is essentially incomprehensible, even though it appears trivial after 2-3 readings.  There is a broken reference in section 4.2.  Section 4.4: "The configuration set of target F M is exactly the cross-product of source FMs' configuration sets, so the semantics of mergin operation is preserved. " It is not a cross product but your circled-cross-product as defined earlier - cross-product (or cartesian product) would give a set of pairs of features as configuration, and this is not what your operator gives.  The definiton of the FS function is extremely well hidden in Table 3. It should be introduced in the text (either in section 2 or in section 3).  The rules for constraints are not well explained. For instance, my intuition is that 3 and 4 should we swapped. Perhaps I am wrong, but the authors do not care to rationalize the rules. Presentation: rules 3 and 4 should explicitly state that A (respectively B) are in FS(Source 1) - otherwise the table has to be interpreted sequentially (or is nondeterministic), because the conditions overlaps with rule 1.  \*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*  Third reviewer's review:           >>> Summary of the submission <<<  This paper proposes an approach to merge multiple feature models into one. At the preprocessing phase of this approach, the SPL developer are required to add rich-refinement types (decomposition, specialization and characterization) into the source feature models. Then the target feature model is built based on three set of rules: how to merge unique features, how to merge common features and how to merge constraints. The merging follows the cross-product semantics so that the configurations of the target feature model will be the combination of all the valid configurations of the source feature models. The author also discusses how they handle the issues about renaming features and parent-incompatibility in the paper. Comparisons with other existing approaches and a preliminary evaluation of the approach are also given.           >>> Evaluation <<<  + The paper is in general well written and easy to read.      + Reasonable manual step to add rich-refinement types into the source feature models, which makes it possible to define a set of merging rules.      + The construction of the target feature model is based on a set of rules which does not involve computations of exponential complexity. This may indicate acceptable scalability of the approach.      - The introduction part is not clear enough, e.g. the concepts of cross-product semantics and rich-refinement types are mentioned without giving the definitions first.      - Critical reference missing. In the section 4.2, instead of referring to the authors' previous work on the rich-refinement types, "[Error! Reference source not found.]" is shown.      - Not enough information on the evaluation part. It would be nice to refer to a repository where all the source feature models and target models in the experiments are made available.  \*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\*=--=\* | | |