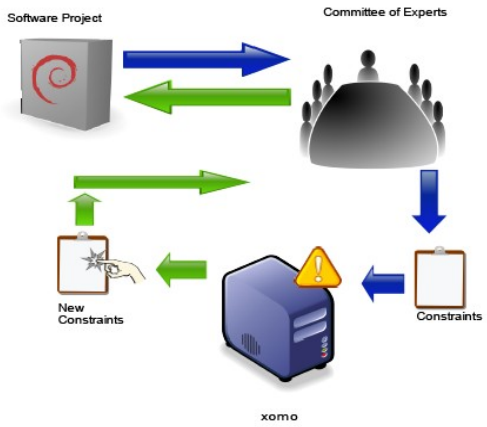

XOMO calibration studies: Sensitivity to change and data perturbation

Oussama El-Rawas

Lane Department of Computer Science and Electrical Engineering, West Virginia University, USA, oelrawas@mix.wvu.edu

Tim Menzies

Lane Department of Computer Science and Electrical Engineering, West Virginia University, USA, tim@menzies.us

Abstract Previously in [1], we sought an AI agent that can negotiate the trade space within the design of modern complex systems such as NASA's new CEV/CLV rockets. In the process of doing that, we tried to assess the merits of using TAR3 treatment learning for such an agent. We found that it provided succinct recommendations that positively affected the development effort when it comes to reducing effort, defects, and threats to the project plan.

In this paper, we show that such recommendations produced by TAR3, in the larger scheme that is XOMO, are stable across separate runs under different conditions. We also show that they are independent of the Cocomo-II and Coqualmo calibrations, causing very little variation upon making changes in these models.

More generally, the conclusion of this work is that we can offer more control to software managers earlier in the life cycle, despite large uncertainties in the domain.

Acknowledgments This research was conducted at West Virginia University under NASA sub-contract project 100005549, task 5g, award 1002193r. All software discussed here is available from http://unbox.org/wisp/ under the GNU Public License version 2: see www.gnu.org/copyleft/gpl.html

Table of Contents

Motivation

Many software engineering models and methods have been developed in order to help in the management of software projects. Aiming to reduce development costs, software defects and project threats, these software engineering models are used by committees of software engineers to help decide what to work on next. The issue is that there are many parameters in these models, and modifying one parameter seems arguably the same as modifying another parameter.

So how should committees decide what to do next? This is where XOMO [2] (X-O-mow) comes into the picture. Based on constraints already determined for the software project, and on the models used, XOMO uses a combination Monte Carlo simulations, discretizers (in this case, "Bore" which is briefly described below), and Treatment learning using TAR3 [3]. The results are generated treatments that promise to produce the largest decrease in effort, defects, and threats (corresponding to the Models Cocomo-II, Coqualmo, and the Madachy Threat model [4]). Illustration 1 is one example showing a graph describing how XOMO influences the aforementioned characteristics of a software project.

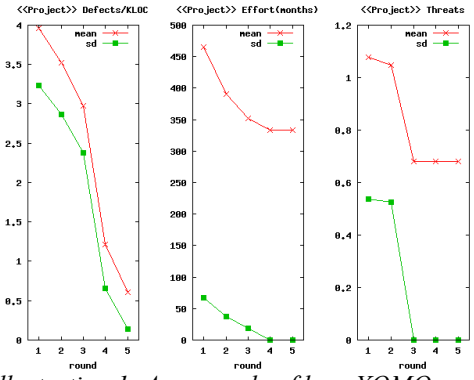

Illustration 1: An example of how XOMO decreases cost, defects, and threats for a certain case

Experimental test bed

All the experiments were conducted on local machines using a copy of a *java* tool that allows the use and configuration of XOMO. This tool, known as XOMOJ, is a basic prototype/proof of concept for the implementation of a tool that would use XOMO in the background while presenting the user with a simple user interface that allows the user to manipulate the running propeties of XOMO. This tool also uses shell scripts in the background to get the results of the XOMO runs, allowing these results to be visualized by using *gnuplot*, and also allowing the runs to be logged in a predefined file structure. The test machines were running *Ubuntu 6.06 LTS*.

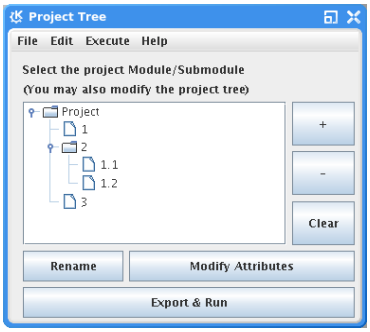

Illustration 2: XOMOJ Main interface

Note that this tool is available in wisp (http://unbox.org/wisp/branches/xomoj/ous/xomojGui/) in the form of a jar executable file. This tool is still in beta form, and lacks support for some feature such as determining ranges for attributes. To do so, manual changes needed to be done on default ranges of the system attributes in the defaults.dat file which is shown in Illustration 3. Also note this tool will be superseded by an eclipse plugin that is being developed for this research.

```
1 # constants
2 A     is real 0.1 10.0
3 B     is real 0.1 10.0
4 ksloc is int 2 10000
5
6 automated_analysis         is int 1 6
7 peer_reviews               is int 1 6
8 execution_testing_and_tools is int 1 6
9
10 # scale factors
11 prec  is int 1 5
12 flex  is int 1 5
13 resl  is int 1 5
14 team  is int 1 5
15 pmat  is int 1 5
16
17 # effort multipliers
18 time  is int 3 6
19 stor  is int 3 6
20 data  is int 2 5
21 pvol  is int 2 5
22 ruse  is int 1 5
23 rely  is int 1 5
24 docu  is int 1 5
25 acap  is int 1 5
26 pcap  is int 1 5
27 pcon  is int 1 5
28 aexp  is int 1 5
29 plex  is int 1 5
30 ltex  is int 1 5
31 tool  is int 1 5
32 sced  is int 1 5
33 cplx  is int 1 6
34 site  is int 1 6
```

Illustration 3: The default project ranges

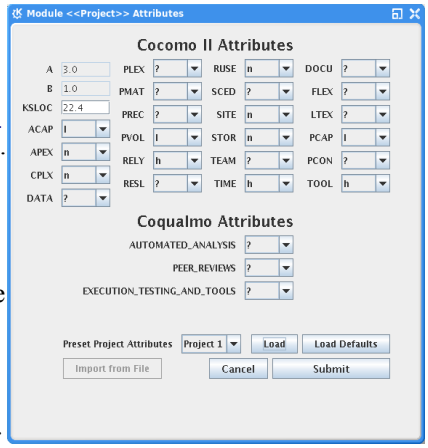

Illustration 4: XOMOJ Attribute modifying interface.

For all the following experiments, one run of the tool refers to running XOMO 5 times in a loop. Each time in the XOMO loop the best treatment is taken and added to the case constraints, and XOMO loops again using the new case constraints to produce incrementally better constraints for the software project, reducing defects, effort and threats for the software project.

Apart from the Monte Carlo generator, XOMO has three main components. The first is the Model, or rather the set of models used in XOMO. These models are used in defining the projects and evaluating them. In this case the models are the

aforementioned Cocomo-II, Coqualmo, and the Threat models. the second component is the discretizer, which in this case is "Bore". Bore stands for Best Or REst, and what it aims to do is to make a discrete two class problem out of a continuous multi-class problem. Basically, it looks at scores of individual samples. It then classifies a previously specified percentage of them that are closest to a "sweet spot" as best. The other samples form the second class, which is rest. The third component is the treatment learner TAR3. This can be described as a minimal contrast set learner .The For more details about XOMO, please refer to [1] at http://menzies.us/pdf/06xomo202.pdf.

```
1 system with ares
2 ksloc just  75 125
3 prec just 3 5
4 flex = 3
5 resl = 4
6 team = 3
7 pmat just 4 5
8 rely = 5
9 data = 4
10 cplx = 4
11 ruse = 4
12 docu just 3 4
13 time = 3
14 stor = 3
15 pvol = 3
16 acap = 4
17 pcap = 3
18 pcon = 3
19 aexp = 4
20 plex = 4
21 ltex just 2 5
22 tool = 5
23 site = 6
24 sced just 2 4
```

Illustration 5: Case1 – Ares ranges

```
1 system with kc1
2 A = 2.94
3 B = 0.91
4 acap just 2 3
5 aexp just 2 3
6 cplx  just 5 6
7 data = 3
8 docu just 2 4
9 ksloc just  75 125
10 ltex just 2 4
11 pcap = 3
12 pcon just 2 3
13 plex = 3
14 pmat just 1 4
15 prec just 1 2
16 pvol = 2
17 rely = 5
18 resl just 1 3
19 ruse  just 2 4
20 sced just 1 3
21 site = 3
22 stor just 3 5
23 team just 2 3
24 tool just 2 3
25 flex just 2 5
```

Illustration 6: Case2 - KC1

The Cases

The cases presented in this paper are defined by setting the ranges/values of the Cocomo-II and Coqualmo attributes of that case, provided that these ranges/values are known for this software project case. These range/value definitions override the default ranges for the models shown in Illustration 3, where "int" indicates the use of whole values, and "real" indicates the use of real numbers, possibly with decimal values. Note that for the int values each number corresponds to an attribute value in the following manner: 1=very low (vl), 2=low(l), 3=nominal(n), 4=high(h), 5=very high(vh) and 6=extremely high(xh). As the amount of attributes with set values increases, the software project becomes more defined, and therefore more constrained in terms of the aforementioned models. This is usually the case for mature software projects. However, when there aren't many set attribute values, and when ranges are more common, the project is said to be under constrained. This is usually the case for new, less mature projects.

There are two cases that are used in the experiments included in this paper, both of which are NASA projects. The first being the Ares project, whose ranges are defined in Illustration 5. The keyword "just" indicates a range of values for the corresponding attribute. As it is clear from Illustration 5, this project has many set values for its attributes, meaning that it is a well constrained and mature project. The second case is from the project "KC1", which is an experimental NASA space plane. The ranges for that project are defined in Illustration 6. Note that there are few set values, and that most of the attribute constraints are in the form of ranges. This leads to the conclusion that this project is yet to mature, and hence can be described as under constrained.

So for Case1, we have:

- Many attributes that already set.
- Very few attributes that are given ranged restrictions.

Hence we can call this case the *mature* project case. for Case2 on the other hand, we have:

- Few attributes that are set.
- Most restrictions are in the form of ranges.

Hence we can call this case the *fledgling* project case.

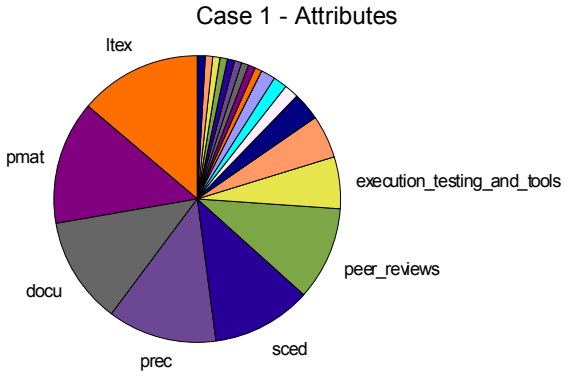

Illustration 7: The results of XOMO for Case 1 over 15 runs for A=3.0 and B=1.0. The dominant attributes look to be ltex, pmat, docu, prec, sced, peer_reviews and execution_and_testing_tools.

	Min	*Max*	*Mean*
A	3.72	9.18	5.97
B	0.85	1.09	0.96

Table 1: The values used for A and B in the Cocomo-II model in the various experiments.

Base Experiments

These base experiments where conducted on the two aforementioned cases: the first being an initially more constrained case, and the second being under constrained in terms of the initial values given for the attributes of the software project.

The initial constraints for these two cases are presented in Illustration 5 and Illustration 6. Note that for the base

experiments, the values for A and B used were set to a constant 3.0 and 1.0 respectively irrespective of the case constraints. For this experiment, XOMO was run 15 times for each case, and the results from all the runs were then documented and organized into separate pie charts in order to get a view of the dominant attributes that XOMO concentrating on. Also the categories of the attributes that XOMO is concentrating on was also documented. There are 6 categories: Scale Drivers, Product attributes, Platform attributes, Personnel attributes, Project attributes and Defect Elimination schemes. The first five categories are taken from the paper [5]. The Last is added by us and includes the defect removal methods included in the Coqualmo model used in XOMO, which are automated_analysis, peer_reviews, and execution_and_testing_tools.

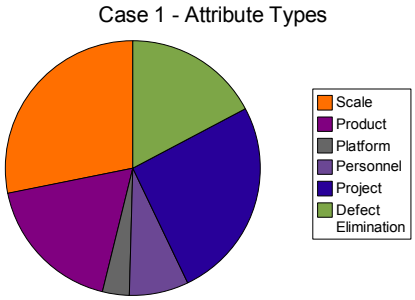

Illustration 8: The types of attributes that were chosen by XOMO for the experiment corresponding to Illustration 7. Notice the low amount of personnel and platform attributes.

The Illustrations 7 and 8 show the base experimental results for Case 1, which as mentioned before is the more constrained case. The first pie chart shows the plot of the frequencies of all the attributes that XOMO chooses to change within all 15 runs. The larger the slice in the chart, the larger the frequency that XOMO is picking the corresponding attribute.

We define a dominant attribute as one that appears at least 50% of the time. In this case, the *dominant attributes* are ltex, pmat, docu, prec, sced and peer_reviews. Execution_testing_and _tools was a bit short of being a dominant attribute.

Looking at the Attribute Types chart for case 1, we notice that the amount of personnel and platform attributes that were suggested by XOMO to be changed were small by comparison to the other attribute types. This suggests that for constrained projects that have already become very well defined for whatever reason, we need not be overly concerned with modifying personnel and platform data related to that project, but rather that we should concentrate on other aspects of the project that are more related to the final product, such as defect elimination. This makes sense, and confirms the results presented in [1].

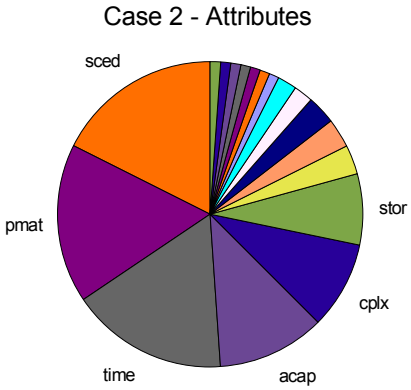

Illustration 9: The results of XOMO for Case 1 over 15 runs for A=3.0 and B=1.0. The dominant attributes look to be sced, pmat, time, acap, cplx and stor.

The illustrations 9 and 10 are the same charts described above, except they pertain to the results that have to do with Case 2. The dominant attributes for this case are the following: sced, pmat, time, acap, cplx, and stor. As for the types of attributes that XOMO is concentrating on, a much larger part of the second chart pertains to the attributes of type platform and personnel, while lacking any mention of any defect elimination attributes. As mentioned before, case 2 is the less constrained, *fledgling* project case. This would suggest that when the software project is not yet well defined, which is usually at the beginning of the software project life cycle, that defect elimination schemes are totally useless, and that more initial emphasis should be placed on process related attributes, i.e. platform and personnel. This result also confirms the conclusions drawn about this same case in [1].

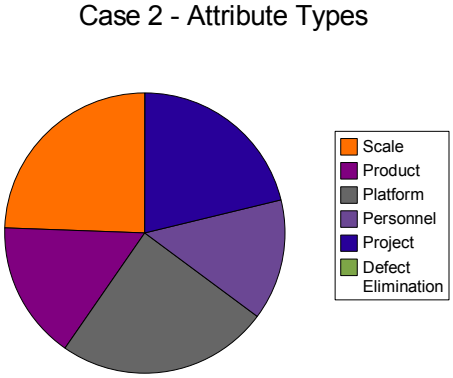

Illustration 10: The types of attributes that were chosen by XOMO for the experiment corresponding to Illustration 9. Notice the absence of any recommendations for defect elimination and the larger concentration on platform attributes.

Another observation to mention here has to do with the stability of the XOMO conclusions. It was previously observed that there are differences in the attributes chosen between the different runs of XOMO for the same case, causing concern about unstable conclusions. However, the chart addressing the attributes chosen for cases 1 and 2 clearly show that there is some kind of

stability across the different runs, where some attributes are being emphasized by XOMO, and others are either rarely mentioned or not mentioned at all. This provides us with what we can call stable theories across the runs. This observation could be used for future work in order to provide more stability to XOMO when it comes to its suggestions.

Case 1 AB Max	vl	l	n	h	vh	xh	total
ltex					12		12
sced			3	7			10
docu			10				10
prec					9		9
peer_reviews			1			8	9
pmat					10		10
execution_testing_and_tools						6	6

Table 2: This table shows the variation in the values of the most frequent attributes chosen by XOMO for the indicated case. Note how there is little variation in these values.

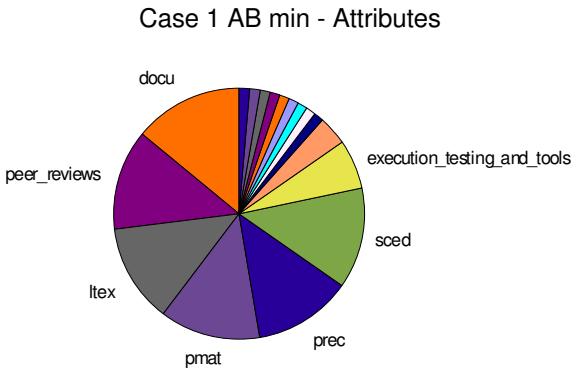

Illustration 11: The results of XOMO for Case 1 over 10 runs for A=3.72 and B=0.85. The dominant attributes look to be docu, peer_reviews, ltex, pmat, prec, sced and execution_test_and_tools. This is identical to the results from Illustration 7.

Varying A/B Experiments

In this section we will be investigating the sensitivity of the results presented by XOMO on the A and B factors that are constants in the Cocomo-II model. These constants are used by Cocomo-II to compute the effort required for a software project. They are usually calibrated based on historical data form other projects. Three different combinations of A and B where used: the maximum, minimum, and mean values for both. The values used are in Table 1.

These values were generated using Linear Calibration (LC) on the *nasa93.csv* dataset available to us. For each combination of A and B in each case, XOMO was run 10 times. The results for case 1 are in Illustrations 11, 12, 24, 25, 26 and 27[1].

Case 1 AB min	vl	l	n	h	vh	xh	total
ltex					10		10
sced			2	8			10
docu			11				11
prec				1	9		10
peer_reviews						10	10
pmat					10		10
execution_testing_and_tools						5	5

Table 3: This table shows the variation in the values of the most frequent attributes chosen by XOMO for the indicated case. Note how there is little variation in these values.

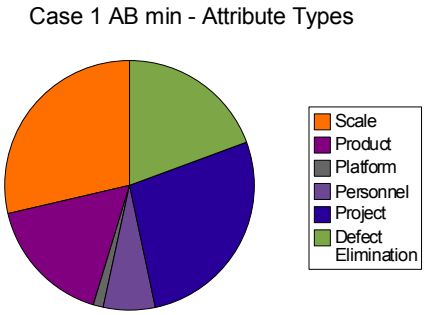

Illustration 12: The types of attributes that were chosen by XOMO for the experiment corresponding to Illustration 11. Notice the low amount of personnel and platform attributes, just as in Illustration 8.

Case 1 AB mean	vl	l	n	h	vh	xh	total
ltex					10		10
sced			3	6			9
docu			10				10
prec					10		10
peer_reviews						10	10
pmat					10		10
execution_testing_and_tools					1	6	7

Table 4: This table shows the variation in the values of the most frequent attributes chosen by XOMO for the indicated case. Note how there is little variation in these values.

As can be seen in the charts presented for Case 1, the results are very similar across all values of A and B, maintaining the same dominant attributes and therefore very similar attribute type distribution. Looking at these results, we not only notice that we have stable results that are independent of A and B, rather we can also clearly see similar results when compared to our base experimental results. This shows that results of XOMO are seemingly independent of the values of A and B, making XOMO not only stable, but also robust when it comes to dealing with badly calibrated A and B values for the Cocomo-II effort model.

1 Illustrations 24 to 35 are in the Appendix

Now that we've shown stability across all values of A and B for case 1 when it comes to XOMO choosing which attributes to modify, we would like to investigate whether there is any instability when it comes to values that XOMO suggests for these attributes.

Case 2 AB min	vl	l	n	h	vh	xh	total
time			9				*9*
sced			10				*10*
pmat			2	8			*10*
cplx					1	4	*5*
acap			6				*6*

Table 5: This table shows the variation in the values of the most frequent attributes chosen by XOMO for the indicated case. Note how there is little variation in these values.

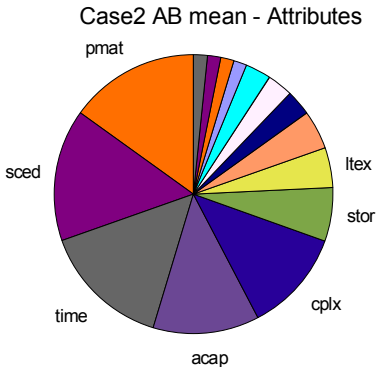

Illustration 13: The results of XOMO for Case 2 over 10 runs for A=5.97 and B=0.96. The dominant attributes are pmat, sced, time, acap and cplx. This is very close to the results from Illustration 9, with stor appearing just short of 50% of the time.

The three tables 2, 3 and 4 document the frequencies of the dominant attributes chosen by XOMO and the values chosen. Note that vl = very low, l =low, n = nominal, h = high, vh = very high and xh = extremely high. Also note the reason why some attributes frequencies exceed the number of runs is because sometimes these attributes are mentioned more than one within a single XOMO run. Looking at these tables, the only attribute that varies significantly seems to be sced, which varies between nominal and high. This seems like an issue at first, however

Case 2 AB mean	vl	l	n	h	vh	xh	total
time			9				*9*
sced			10				*10*
pmat				9			*9*
cplx					1	7	*8*
acap			7				*7*

Table 6: This table shows the variation in the values of the most frequent attributes chosen by XOMO for the indicated case. Note how there is little variation in these values.

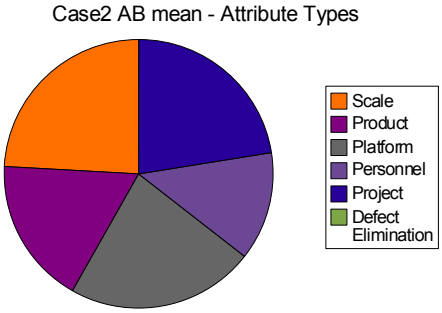

Illustration 14: The types of attributes that were chosen by XOMO for the experiment corresponding to Illustration 13. Notice the absence of any recommendations for defect elimination and the larger concentration on platform attributes.

taking a closer look at the Cocomo-II model [4] reveals that this isn't an issue since the values for sced in nominal and high are the same (they are both 1.0). So overall, we observe not only stability of the attributes chosen, but also stability pertaining to the values chosen for these attributes in well constrained situations.

Next the charts for case 2 will be presented. The Running conditions are the same at that of the case 1.

Case 2 AB max	vl	l	n	h	vh	xh	total
time			8	3			*11*
sced			10				*10*
pmat			2	7			*9*
cplx					10		*10*
acap			9				*9*

Table 7: This table shows the variation in the values of the most frequent attributes chosen by XOMO for the indicated case. Note how there is little variation in these values.

Again, case 2 is the less constrained case here. That being said, the results across the values of A and B are very much the same, displaying very similar dominant attributes and attribute type distribution, both of which highly resemble the base experimental results. However, it should be mentioned that compared with case 1, we have greater variance since non-dominant attributes take up a larger portion of the pie charts in Illustrations 13, 28, and 29 indicating less stability.

Tables 5, 6, and 7 are the same as tables 2, 3 and 4 except they correspond to the results of case 2. Looking at these tables, we can see that there isn't a significant amount of variation when it comes to the values recommended for the attributes chosen. We do notice however that for cplx there is a relatively sharp change in the overall amount of times that it turns out in the XOMO recommendations. On the other hand though, it still qualifies as a dominant attribute across the values of A and B.

Having shown that the output recommendations of XOMO aren't dependent on A nor B, we will move on to show that it is also independent of the values in the Cocomo-II and Coqualmo tables. Before doing that, we will present a brief description of how we modified the model values.

Model Calibration Independence

The Models

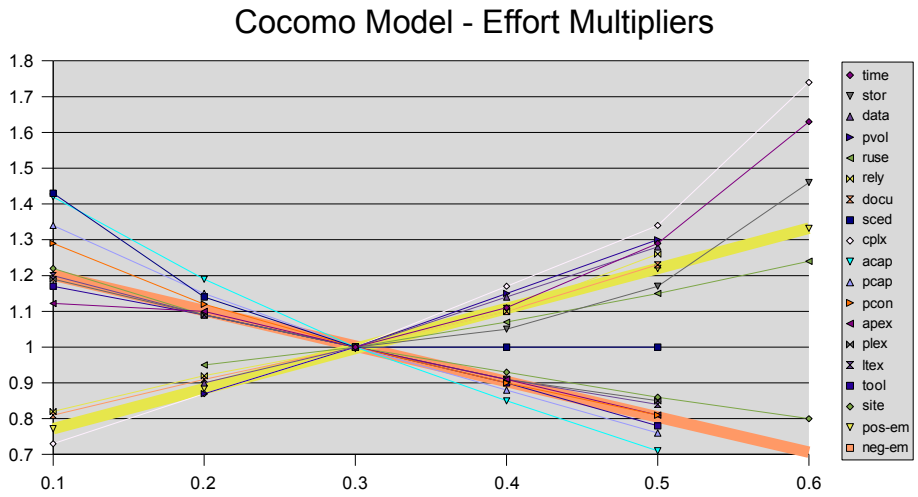

Illustration 15: This is the graph of the effort multipliers of Cocomo-II. The two thick lines represent the average slopes of the values: one for those with positive slopes, and the other for those with negative slopes.

There are three models that are curently used in XOMO: Cocomo-II, Coqualmo, and the Threat model [4]. For the purpose of these experiments, only the first two models were modified within XOMO to test XOMO's sensitivity to these models. For the Cocomo-II model, the attributes are divided into two main categories: The Effort multipliers and the Scale factors. The above Illustration is a graph of all the Effort multipliers values. Note that the x-axis corresponds to the value of the attributes, 0.1 being vl (very low) and 0.6 being xh (extremely high). As it is clear in the graph, the Effort multipliers can be divided into two categories: Those with a positive slope, and those with a negative slope. For each of these subcategories, an average slope is derived from the original model, and a line is made to represent the whole subcategory. In the graph above, these two lines are the thick lines labeled *pos-em* and *neg-em*. The values for these lines replace all the values of the corresponding attributes from which they were derived. The same is done for the Scale Factors. This causes the Cocomo-II model within XOMO to be reduced into simply three lines.

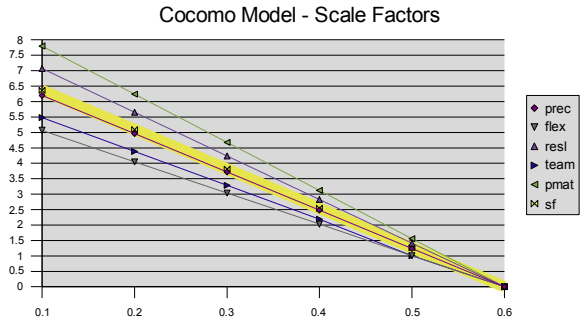

Illustration 17: This graph shows the points that are the standard values of the Cocomo-II Scale factors. The thick line represents the line that has a slope that is the average of the slopes of all the other lines.

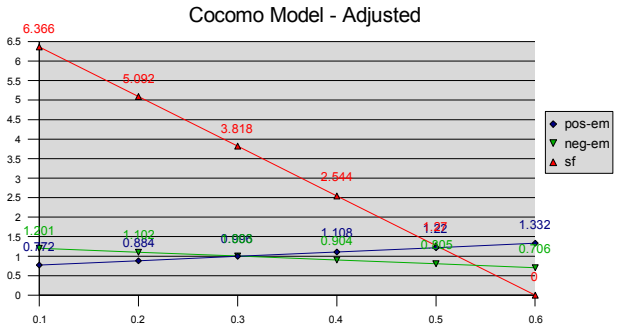

Illustration 16: This Graph represents the simplified Cocomo-II model made of the 3 thick lines shown in Illustrations 15 and 17. The values indicated on the lines are the values used in the Adjusted Cocomo-II model.

The same procedure is also followed with the Coqualmo model, where it is reduced into three lines: the positive sloped line, the negative sloped line, and the line corresponding to attributes that don't vary. This had to be done three times overall for Coqlmo: for each of the coding, requirements and design tables. The illustrations 16, 19, 34, 35 show the models and the

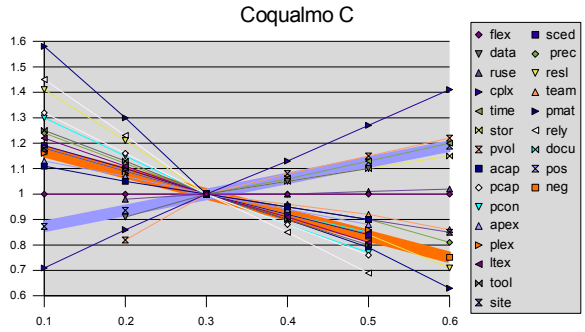

Illustration 18: The plot of the attribute values in Coqualmo – C (Coding), and the corresponding average lines.

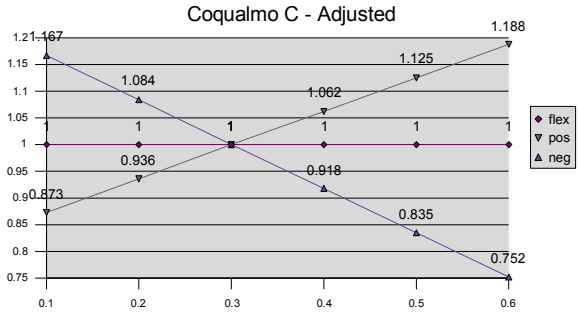

Illustration 19: The reduced Coqualmo – C (Coding) model, with the values indicated on the graph (0.1 = vl, ..., 0.6 = xh)

graphs showing the new reduced models.

The Experiment

Using the above indicated modified models in XOMO, and using the mean values for A and B indicated in Table 1, we ran XOMO again on both cases 1 and 2 in order to see how sensitive XOMO was to the calibrations of these models. For each of the cases, XOMO was run 10 times and the results were documented and graphed in the charts presented below for both cases 1 and 2.

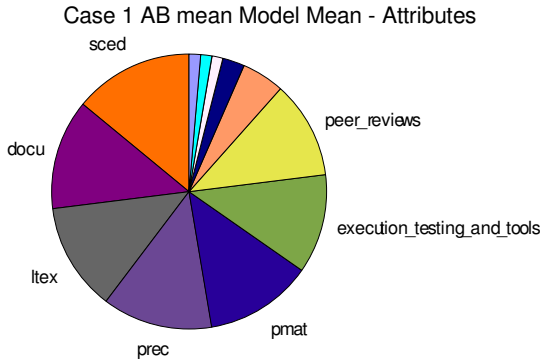

Illustration 20: The results from XOMO for Case 1 over 10 runs when using the mean values of A and B and the simplified Cocomo-II and Coqualmo models that result from using the mean slopes in these models. The dominant attributes are sced, docu, ltex, prec, pmat, execution_testing_and_tools and peer_reviews. These results are identical to those in the base experiment.

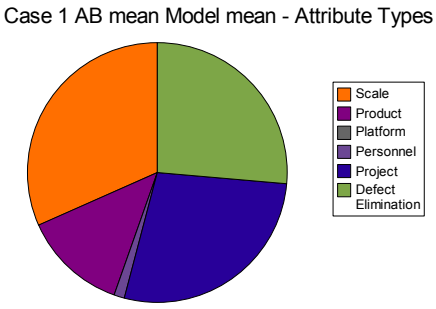

Illustration 21: The types of attributes that were chosen by XOMO for the experiment corresponding to Illustration 20. Notice the low amount of personnel and lack of platform attributes.

Illustrations 20 and 21 show the results for case 1 for the simplified versions of Cocomo-II and Coqualmo. Note that results in terms of dominant attributes are identical to those in the base experiment, and that there is a low amount of/lack of personnel and platform attributes chosen. An interesting observation to note is that modifying the model somehow increased the stability of XOMO results for this case, evidenced by the larger proportion that the dominant attributes take in the corresponding results pie chart. This suggests that not only is a constrained project with simpler models produce the same results, but that also improves the stability and "confidence" of the XOMO results and suggestions.

Illustrations 22 and 23 correspond to the results for case 2. Note the results are very similar to the base experiments. Although there isn't much change relating to the stability of XOMO under these conditions compared to the base conditions, this definitely shows that the results, whether for cases 1 or 2, are for the most part independent of the initial calibrations of the Cocomo-II and Coqualmo models used in XOMO.

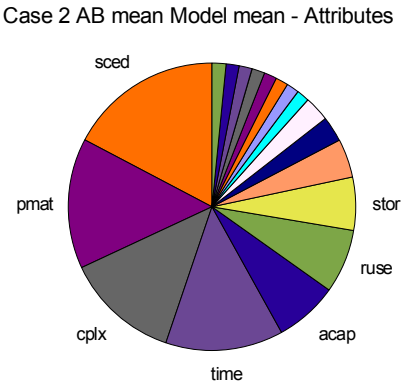

Illustration 22: The results from XOMO for Case 1 over 10 runs when using the mean values of A and B and the simplified Cocomo-II and Coqualmo models that result from using the mean slopes in these models. The dominant attributes are sced, pmat, cplx, time, acap and ruse. These results are very similar to those in the base experiment, with the exception of the inclusion of ruse.

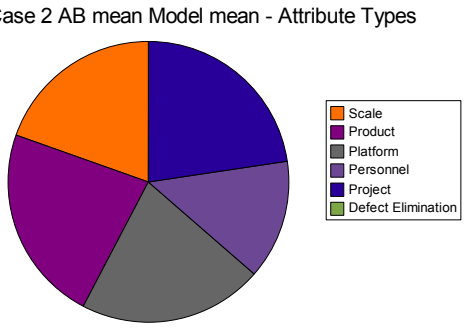

Illustration 23: The types of attributes that were chosen by XOMO for the experiment corresponding to Illustration 22. Notice the absence of any recommendations for defect elimination and the larger concentration on platform attributes.

Conclusion

In this paper, we conducted several experiments with XOMO to investigate its stability on the one hand, and its sensitivity to the values used in the Cocomo-II and Coqualmo models. For all the experiments we used the configuration tool XOMOJ and ran the experiments on data from two nasa projects: Ares, which was the first case, and KC1, the second case. Ares was described as being more constarained, which is characteristic of a mature project. KC1 on the other hand was less mature since it was much less constrained.

Three main experiments were run: the first to establish a baseline and to investigate stability, the second to investigate dependence on the A and B values in the Cocomo-II model, and the third to investigate dependence on the Cocomo-II and Coqualmo table values. As a result of these experiments, were were able to see stability of the XOMO decisions in all the situations. We also saw how the results from changing model values didn't change the conclusions of XOMO drastically. Also, the results presented in [1] were confirmed with respect to the type of attributes that XOMO chose in the two case studies. Ares, the well constrained one, prompted XOMO to look away from platform and personnel attributes and concentrate on other attributes, such as defect elimination schemes. KC1 on the other hand, being the under constrained project, prompted XOMO to totally neglect defect elimination schemes and to concentrate more on process related issues such as platform and process attributes.

Apart from confirming previous results and conclusions, we were also capable of producing results that clearly show that XOMO and the recommendations it is producing is independent of the calibrations of the values in the Cocomo-II and Coqualmo models. This shows that XOMO is not only stable for properly calibrated models, but also that it is stable even for badly calibrated models, producing very similar recommendations. We can also deduce that the more under constrained a project is, the less stable the results of XOMO will be (even though we will still see a definite pattern in the recommendations).

While the results presented above are promising for XOMO, several experiments and issues need to be investigated:

- More experiments need to be done on the independence of XOMO from Model values. More specifically, the slopes of the lines in the new models need to be varied around a point in hinge fashion to see what the effect would be on XOMO's recommendations.
- The cofiguration tool used here is still a very much basic tool. to facilitate future experiments, a more advanced tool needs to be developed. Currently, one such tool that uses the same backend is being developed as an eclipse plugin.
- Enhancing the backend, mainly TAR3 (the treatment learner used) by implementing a boosting scheme geared toward treatment learning to possibly enhance stability.

Bibliography

- 1: T. Menzies, RST milestone 1.1.5.9:Applying trade space analysis to recommend CEV/CLV options for SW capabilities and development of processes and tools, WVU Tech Report, 2006, http://menzies.us/pdf/06xomo202.pdf
- 2: T. Menzies and J. Richardson, Xomo: Understanding development options for autonomy, COCOMO forum, 2005, http://menzies.us/pdf/05xomo_cocomo_forum.pdf
- 3: T.Menzies and Y.Hu, Data mining for very busy people, IEEE Computer, November 2003, http://menzies.us/pdf/03tar2.pdf
- 4: B. Boehm, E. Horowitz, R. Madachy, D. Reifer, B. K. Clark, B. Steece, A. W.Brown, S. Chulani, and C. Abts, Software Cost Estimation with Cocomo II, Prentice Hall, 2000
- 5: T. Menzies and E. Sinsel, Practical large scale what-if queries: Case studies with software risk assessment, IEEE ASE, 2000, http://menzies.us/pdf/00ase.pdf

Appendix

This Appendix includes many illustrations that represent the results of the experiments presented in this paper. They are included here since most of them show similar results for different scenarios, and in order to save space within the paper itself.

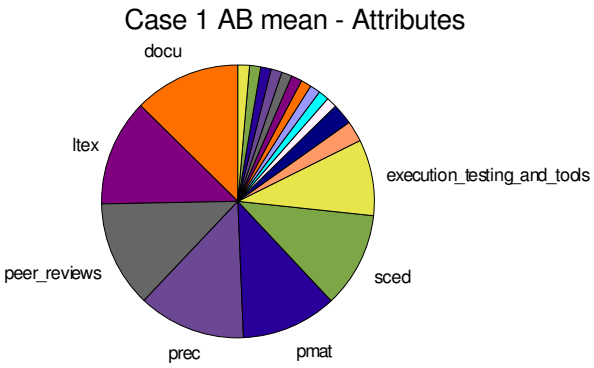

Illustration 25: The results of XOMO for Case 1 over 10 runs for A=5.97 and B=0.96. The dominant attributes look to be docu, ltex, peer_reviews, prec, pmat, sced and execution_test_and_tools. This is identical to the results from Illustration 7.

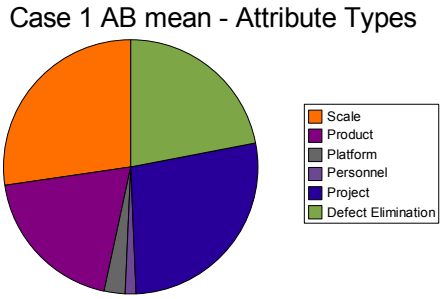

Illustration 24: The types of attributes that were chosen by XOMO for the experiment corresponding to Illustration 25. Notice the low amount of personnel and platform attributes, just as in Illustration 8.

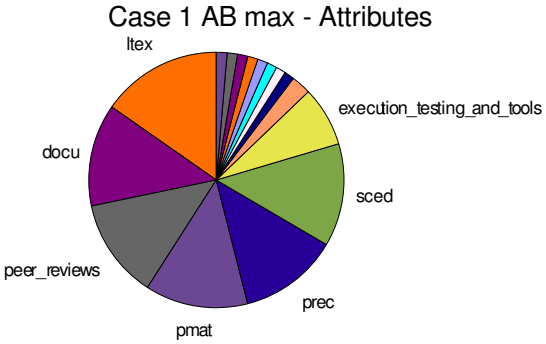

Illustration 27: The results of XOMO for Case 1 over 10 runs for A=9.18 and B=1.09. The dominant attributes look to be ltex, docu, peer_reviews, pmat, prec, sced and execution_test_and_tools. This is identical to the results from Illustration 7.

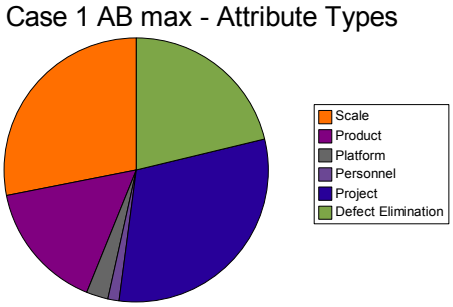

Illustration 26: The types of attributes that were chosen by XOMO for the experiment corresponding to Illustration 27. Notice the low amount of personnel and platform attributes, just as in Illustration 8.

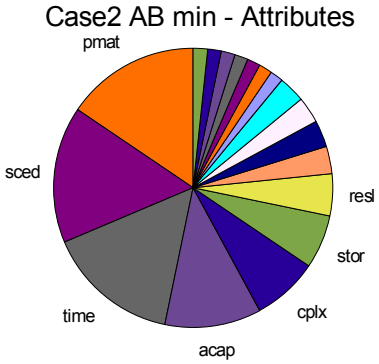

Illustration 28: The results of XOMO for Case 2 over 10 runs for A=3.72 and B=0.85. The dominant attributes are pmat, sced, time, acap and cplx. This is very close to the results from Illustration 9, with stor appearing just short of 50% of the time.

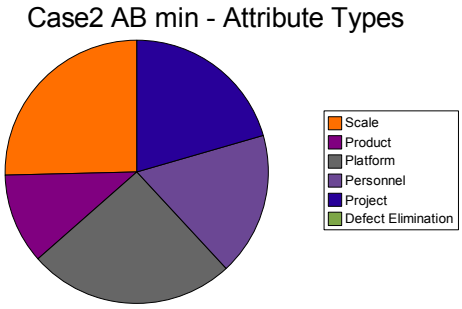

Illustration 32: The types of attributes that were chosen by XOMO for the experiment corresponding to Illustration 28. Notice the absence of any recommendations for defect elimination and the larger concentration on platform attributes.

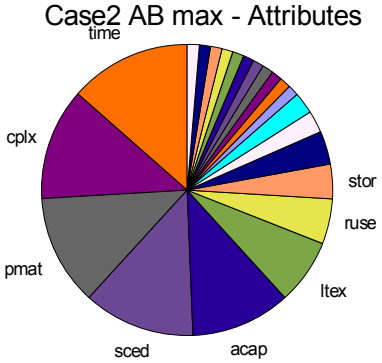

Illustration 29: The results of XOMO for Case 2 over 10 runs for A=9.18 and B=1.09. The dominant attributes are time, cplx, pmat, sced, acap and ltex. This is very close to the results from Illustration 9, with ltex being one of the less dominant attributes.

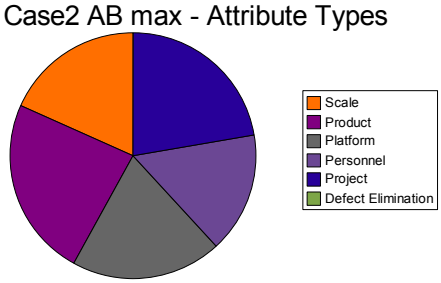

Illustration 33: The types of attributes that were chosen by XOMO for the experiment corresponding to Illustration 29. Notice the absence of any recommendations for defect elimination and the larger concentration on platform attributes.

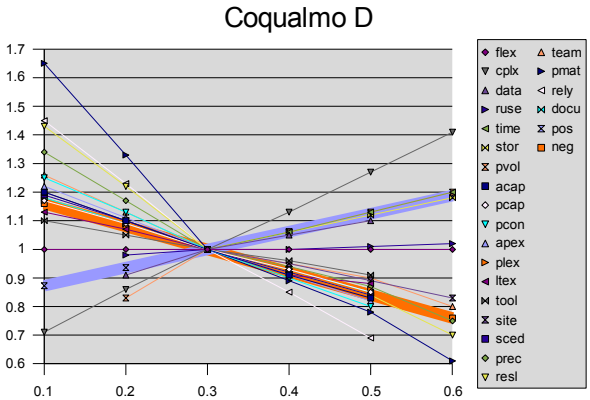

Illustration 30: The plot of the attribute values in Coqualmo – D (Design), and the corresponding average lines.

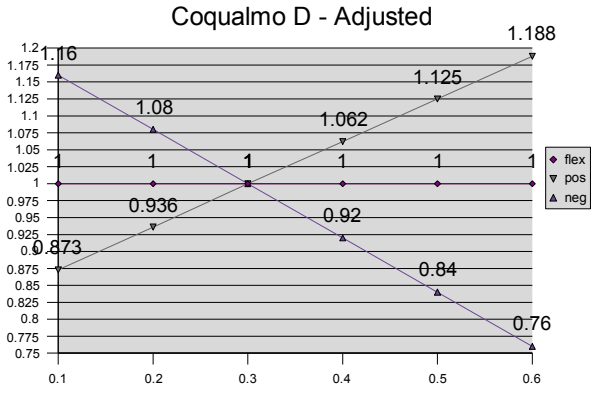

Illustration 34: The reduced Coqualmo – D (Design) model, with the values indicated on the graph (0.1 = vl, ..., 0.6 = xh)

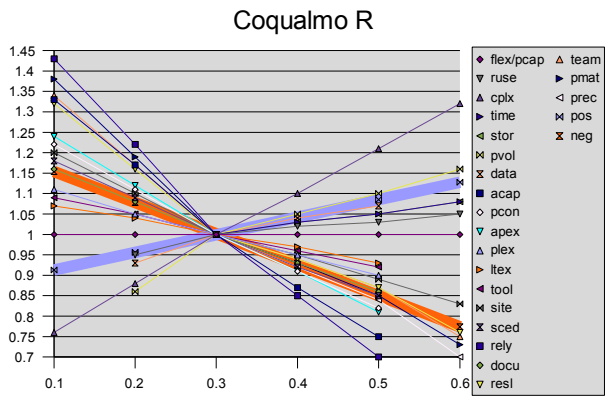

Illustration 31: The plot of the attribute values in Coqualmo – R (Requirements), and the corresponding average lines.

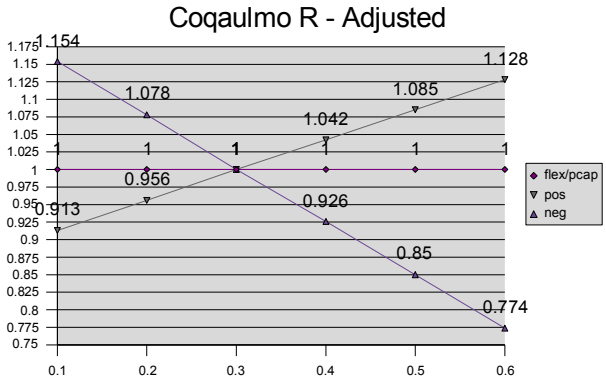

Illustration 35: The reduced Coqualmo – R (Requirements) model, with the values indicated on the graph (0.1 = vl, ..., 0.6 = xh)