SCHOOL OF COMPUTING

Postgraduate Conference 2010

Technical Report 02-10

Edited by
Laurence Hellyer, Carl G. Ritson and Jonathan Simpson

University of Kent | Computing

Introduction

It is our pleasure to introduce you to the proceedings of the School of Computing's annual Postgraduate Conference 2010. Students were invited to submit two page papers and posters for peer review, received reviews of their papers and posters and were asked to produce camera-ready copy. We are pleased that so many students contributed and took part in the process.

The quality of submissions was universally high and we feel it reflects the School's vibrant and active postgraduate research community. We are delighted to see a wide range of research groups represented in the proceedings.

We are indebted to many people for their help and advice including Sally Fincher, Sandra Shine, Sonnary Var and Mark Wheadon, whose valued input and assistance has made arranging this conference possible in such a short space of time. We look forward to enjoying the continuation of this series next year, by other intrepid souls.

Laurence Hellyer
Carl G. Ritson
Jon Simpson
(Editors.)

April 2010.

Contents

Modelling fim in Escherichia coli

Patrick de Vries
School of Computing
University of Kent
Canterbury, UK, CT2 7NF
pd79@kent.ac.uk

ABSTRACT

Trying to figure out how a biological process works can be tricky as there are so many different elements involved. Targeted mutations to the DNA can be used to disable binding or transcription of a certain protein, but this could have an effect on other parts. Modelling this using an agent-based model can be a good solution as long as you have reliable data from experiments. This includes binding probabilities of protein. However such data may not always be available or accurate enough. The subject of this paper is to show the possibility of testing a hypothesis and simultaneously get a good idea of the parameters with an agent-based model using to a genetic algorithm to come up with the parameters for the model.

1. INTRODUCTION

Fimbriae are hair-like attachments that *E.coli* bacteria can grow. With these fimbriae they can attach themselves to host cells and can even penetrate them. Because of their ability to penetrate cells, infections with *E.coli* bacteria are very hard to treat, so it is imperative we learn more about the way the fimbriae are regulated.

The main method of researching the processes within a bacterial cell is by making focused mutations of the DNA. However, a mutation can have further effects within the cell, then just the process focussed upon. therefore an effect attributed to a certain DNA fragment or protein can in fact have an impact via a different mechanism. One can try using a computer model to simulate the process, but for this one needs additional parameters to work to model. This paper is to show an attempt in testing the model with a minimum amount of experimental data, and is a first in combining agent-based modelling with genetic algorithms.

Here the focus will be on the hypothesis that the protein H-NS will act as a repressor for *fimB* expression and the protein SlyA will be acting as an antagoniser of H-NS preventing it from binding to the DNA. Identified are two SlyA binding sites called O_{SA1} and O_{SA2} and a possible third site called O_{SA3} The sites O_{SA1} and O_{SA3} overlap not only with each-other, but also with an H-NS binding site.

H-NS represses *fimB* expression and is antagonised by SlyA. FimB in turn will switch the *fim*-switch (*fimS*) ON which will start the production of fimbriae. While the switch is turned ON FimE will be formed which will stimulate *fimS* to switch OFF.

2. PREVIOUS WORK

Previous attempts have been made at modelling certain aspects of *E.coli* bacteria, for example by means of differential equations [4, 5], where there is focus on the individual cell and the processes within or the focus is on the entire population [3].

These models are very useful for understanding how a single cell operates in terms such as cell growth or nutrient uptake and metabolisation, but when looking at a process which is not continuous, such as cell division, it does not resemble actual bacteria. For this reason we can use agent-based models [9, 11]. These two models had the fortune of having good experimental data available to acquire parameters for there model.

One of the parameters required is binding affinity, and therefore we test the hypothesis by using an agent-based model supported by genetic algorithms.

A reason for using this method is that there is no direct way of measuring binding affinity of the protein SlyA. SlyA has a dissociation constant K_D over the entire DNA of $16 - 24nM$. Normally when doing these gel shifts at different concentrations of the protein for testing its binding affinity clear bands appear for the parts where the protein is bound to the DNA. These bands also should appear at every experiment at roughly the same location as for example LRP as shown by [12]. For unknown reasons gel shifts with SlyA produce irregular banding. The only band at the same location is that of the unassociated DNA. If this were the case then we could have used a similar method as [13] used in their model for calcium-calmodulin interaction.

The genetic algorithm should help in finding the missing parameters, although the goal of this research is not to find these parameters, but use the parameters found as a second test.

There are many interpretations on how *fim* expression is regulated in *Escherichia coli* [10, 12]. A main regulatory process in the expression is down to a fragment of DNA that can be expelled and reinserted in opposite direction [2, 1]. It can be seen as a real [8, 14] switch turning from OFF to ON and back. Also is known that the switch is regulated by the protein FimB and FimE, where FimB is expected to turn the switch from OFF to ON and FimE favours the OFF position.

It is not clear how the processes work that regulate *fimB* expression. An idea is that H-NS (Histone-like nucleoid-structuring) protein [7] represses *fimB* expression and SlyA — a protein first discovered in *Salmonella* — antagonises H-NS [6]. These are just a fraction of the several different regulatory binding sites for FimB.

3. MATERIALS AND METHODS

To test the hypothesis 5 models were created (table 1). The computational model starts out with generation a population of solutions for the parameters we call genes. The solutions are tested and get a fitness assigned. This is stored in an output-file. The test is done by feeding the solutions into a test environment, where it will be assigned to the behaviour of the individual cells.

The model is to be fit on experimental data obtained from replacement mutation experiments. These mutations include RM39, RM40

Model	Summary
1	Basic model
2	As model 1, but no effect of RM39 and RM42 on H-NS
3	As model 1, but effect of O_{SA1} and O_{SA3} are forced to be 50% or more
4	Combination of the models 2 and 3
5	Effects of binding of SlyA to the different sites is separate from effect of replacement mutations + inclusion of SlyA mutation

Table 1: Summary of differences in the 5 models.

and RM42, where RM39 replaces O_{SA1}, RM40 replaces O_{SA2} and RM42 replaces O_{SA3}. RM39 and RM42 have a direct effect on the binding of H-NS. In wild type background the absence of H-NS is tested and as with the other experiments either with or without SlyA present.

The first generations of genes/parameters are randomly created. After being tested for their fitness, the best solution is kept for the next generations and the rest of the solutions are generated by means of cross-over. The candidates for cross-over are selected by tournament selection, where from a selection of four solutions the two strongest are mixed. The new solutions are then subjected to random mutation, where one of the genes is altered.

For every test a new environment is created in which a fixed number of afimbriate cells start out. Each individual run is saved as a text file, containing the number of fimbriate and afimbriate cells at each iteration.

The parameters found are part of a second test. Firstly the model should go towards an optimum and secondly, the parameters produced should be scrutinised with what we know about the process from biology.

The basic model includes effects of binding of O_{SA1}, O_{SA2} and O_{SA3} and the combinations of 1+2 and 2+3 on H-NS binding, as does RM39 and RM42. The effect of the replacement mutations is assumed to be the same as the effect as O_{SA1} or O_{SA3} binding to the DNA. The difference between the 5 different models is shown in table 1. In total there are 14 parameters or genes in the genetic algorithm.

4. RESULTS

The genetic algorithm goes in most circumstances to an optimum. For models 1, 2, 4 and 5 the final results compare very favourably with the experimental results. Still, the acquired parameters have some problems with their biological validity.

The parameters show in model 1, 2 and 4 that both O_{SA1} and O_{SA3} (overlapping with the H-NS site) have no effect on the binding of H-NS, or the effect of H-NS on *fimB* expression. As binding of SlyA and the effect of mutation in these models is assumed to be the same, the effect of RM39 and RM42 is also expected to be zero. This cannot be the case as these two binding sites overlap directly with H-NS and should thus have a direct effect on H-NS.

In model 5 the effect of the mutation and the binding of SlyA is separated. In this case we see an effect of binding of SlyA to both O_{SA1} and O_{SA3}, but the effect of RM39 and RM42 comes out as 0% and 1% respectively. RM40 which has no overlap with the H-NS site is expected to have an effect of 8%.

A further discrepancy is that the *fim* switch is easily switched from ON-to-OFF, but quite difficult to switch from OFF-to-ON. The model produces fimbriate cells in the same ratio as in biological experiments, so maybe the switch behaves like this in real life.

5. DISCUSSION

The validity of the assumptions in two ways, firstly the model has to reach an optimum by genetic algorithms. From this perspective we see that the models 1, 2, 4 and 5 seem to do this. The only model that doesn't go to an optimum is Model 3. The restriction causes it to not go to an optimum, with a least square error (lse) of 10. Model 4 which is based on model 2 and 3 comes closer but has a big variation, where the lse goes down to 2.9. This value is combined with a large variation in results.

The next step is to look at the parameters generated. We can ignore the results for model 3 as it failed to reach an optimum in our first test and also model 4 can be put in doubt, certainly with the rather large errors.

Firstly observing the parameters that say at what concentration of FimB the switch has a 50% probability of going from Off to On and at what concentration the switch has 50% probability of turning Off. In the values obtained here in every model it is quite unlikely for the switch to be in the ON position, or if it does happen only very briefly. We have to note here that in experiments normally is found that only 1% of the cells is in fimbriate state.

The results from model 1 it is implied that neither SlyA binding to the DNA at O_{SA1} or O_{SA3} has any effect on H-NS, nor the mutations of these sites, RM39 or RM42. In Model 5 we see an effect for SlyA binding to O_{SA1} and O_{SA3} but no effect of RM39 and only slight effect of RM42 (G12 and G14). Since these sites overlap directly with H-NS we expect there to be a definite effect.

Model 2 is ignored out of hand even though it has the lowest lse of the five models, but the assumption, like in model 4, that the mutations RM39 and RM42 have no effect on H-NS cannot be accepted as biologically valid.

Since we don't find that the parameters can be seen as biologically valid, this may point to a flaw in the hypothesis. Further testing is necessary in the actual computer model and further research to the correct hypothesis has to be continued. Our estimation is that a further identified H-NS site should be taken into consideration, as well as a possible repressor site located further upstream from the *fimB* site.

Further indications on running the computer model without having SlyA bind to O_{SA3} seem to indicate that this site may not actually be a SlyA binding site. this is backed up by the fact that the O_{SA3} site does not conform to the consensus of the normal SlyA binding site. O_{SA1} and O_{SA2} have a relatively close fit with the consensus for SlyA, but O_{SA3} is a bit off.

A quick test using the same computer model, but preventing SlyA from binding to O_{SA3} seems to indicate that this may be the case, but still doesn't fit perfectly with experimental results, therefore implying that there must be further interaction taking place. There are some small clues towards these other interactions, but further modelling as well as further experiments need to be done to confirm the existence of these other interactions and binding sites.

Recent, but yet to be published research seem to support the idea of just two SlyA binding sites, and seems to go as suggesting four H-NS site, with direct overlap with both of the SlyA sites.

6. REFERENCES

[1] A. M. Adiciptaningrum, I. C. Blomfield, and S. J. Tans. Direct observation of Type 1 fimbrial switching. *EMBO reports*, 10(5):527–532, March 2009.

[2] I. C. Blomfield, D. H. Kulasekara, and B. I. Eisenstein. Integration host factor stimulates both FimB- and FimE-mediated site-specific DNA inversion that controls phase variation of type 1 fimbriae expression in *Escherichia coli*. *Molecular Microbiology*, 23(4):707–717, 1997.

[3] T. Chen, H. L. He, and G. M. Church. Modeling gene expression with differential equations. *Pac Symp Biocomput*, pages 29–40, 1999.

[4] D. Chu and I. C. Blomfield. Orientational control is an efficient control mechanism for phase switching in the *E. coli fim* system. *Journal of Theoretical Biology*, 244(3):541–551, 2007.

[5] D. Chu, J. Roobol, and I. Blomfield. A theoretical interpretation of the transient sialic acid toxicity of a *nanR* mutant of *Escherichia coli*. *Journal of Molecular Biology*, 375:875–889, 2008.

[6] D. Corbett, H. J. Bennet, H. Askar, J. Green, and I. S. Roberts. SlyA and H-NS regulate trascription of the *Escherichia coli* K5 capsule gene cluster, and expression of *slyA* in *Escherichia coli* is temperature dependent, positively autoregulated, and independent of H-NS. *Journal of Biological chemistry*, 282(46):33326–33335, 2007.

[7] R. T. Dame, M. S. Luijsterburg, E. Krin, P. N. Bertin, R. Wagner, and G. J. L. Wuite. DNA bridging: a property shared among H-NS-like proteins. *Journal of Bacteriology*, 187(5):1845–1848, 2005.

[8] T. S. Gardner, C. R. Cantor, and J. J. Collins. Construction of a genetic toggle switch in *Escherichia coli*. *Nature*, 403:339–342, January 2000.

[9] R. Karmakar and I. Bose. Stochastic model of transcription factor-regulated gene expression. *Physical Biology*, 3:200–208, 2006.

[10] P. Klemm. Two regulatory *fim* genes, *fimB* and *fimE*, control the phase variation of type 1 fimbriae in *Escherichia coli*. *The EMBO Journal*, 5(6):1389–1393, 1986.

[11] S. Ramsey, D. Orrell, and H. Bolouri. Dizzy: Stochastic simulation of large-scale genetic regulatory networks. *Journal of Bioinformatics and Computational Biology*, 3:415–436, 2005.

[12] P. L. Roesch and I. C. Blomfield. Leucine alters the interaction of the leucine-responsive regulatory protein (Lrp) with the *fim* switch to stimulate site-specific recombination in *Escherichia coli*. *Molecular Microbiology*, 27(4):751–761, 1998.

[13] N. V. Valeyev, D. G. Bates, P. Heslop-Harrison, I. Postlethwaite, and N. V. Kotov. Elucidating the mechanism of cooperative calcium-calmodulin interactions: a structural systems biology approach. *BMC Systems Biology*, 48(2), 2008.

[14] D. M. Wolf and A. P. Arkin. 15 minutes of *fim*: Control of phase variation in *E.coli*. *OMICS: A Journal of Integrative Biology*, 6(1):91–114, January 2002.

Complexity Analysis of Evolved Programs

Tom Castle and Colin G. Johnson
Computing Laboratory
University of Kent
Canterbury, UK, CT2 7NZ
{tc33, C.G.Johnson} @ kent.ac.uk

ABSTRACT

Program evaluation is typically the most computationally expensive part of evolutionary approaches to automatic programming. This research looks at the feasibility of using complexity metrics as a way of avoiding unnecessary expensive evaluations. Our analysis of program complexity indicates that, for different problems, certain ranges of complexity do give a higher density of highly fit programs.

1. INTRODUCTION

In evolutionary optimisation algorithms it is necessary to evaluate the quality of candidate solutions to determine their chance of survival through to later generations. Where the candidate solutions are programs, such as in Genetic Programming [8], the individuals must be executed, often multiple times. This is an expensive process, particularly for tasks such as image processing problems. Improving the performance of the evaluation step, or reducing the number of evaluations required, will therefore be hugely beneficial to the rate at which the search space can be navigated. The work we present here is a preliminary study into the feasibility of using complexity metrics as part of a system to give such a performance boost.

2. BACKGROUND

The problem of evaluation performance has been tackled with a number of different techniques in the literature. Some approaches focus on parallelising the code [1, 7] or otherwise seeking raw performance gains through inventive uses of hardware, such as GPUs [6]. Others take a less bruteforce approach, by caching subtree fitness [10], or compilation of genomes down to machine code [4].

All these techniques seek to optimise individual evaluations. In contrast, the system we propose in section 5, attempts to avoid evaluation of many programs entirely. This would be achieved by selectively choosing which programs to evaluate according to their complexity score. For this to be possible, a sufficient relationship must exist between the program's fitness and complexity scores. Establishing whether this is the case is the primary purpose of this paper.

Complexity metrics are often used in the field of software measurement to provide a quantifiable measure of software quality. Many different complexity metrics have been proposed [9, 3, 5], primarily with the purposes of identifying software modules that will be difficult to test or maintain, or to hint at the reliability of the code. McCabe's cyclomatic complexity [9] is possibly the most well known of these, and so is the one we consider here.

Cyclomatic complexity is a measure of the number of control paths through a program. For example, a program devoid of conditional statements or loops would have just one execution path and so have a cyclomatic complexity of 1, whereas a program with one 'if' statement would have a score of 2, due to having two independent paths possible. Significantly, the cyclomatic complexity is not merely a measure of program length or depth and is only loosely related to those measures. Throughout the rest of this paper, cyclomatic complexity will be referred to simply as *complexity*.

3. METHOD

It is hoped that this work will be applicable to all evolutionary programming systems but for the purposes of presenting experimental data, this study will use the grammar guided approach outlined by Whigham [11], which he calls Context Free Grammar GP (CFG-GP) as provided by the EpochX GP framework [2]. 100 runs of the algorithm were performed on each of even-five parity, Santa Fe trail and 6-bit multiplexer. The parameters used for each run are outlined in tables 1, 2 and 3. After each generation of each run, the fitness and complexity of all candidate programs in the population were calculated and logged. Standardised fitness is used here, where zero is the best achievable fitness score. If any solutions obtained a zero fitness score then that run was terminated and execution continued with the next run.

Table 1: Even-five parity parameter tableau for CFG-GP

Raw & standardised fitness:	Number of inputs producing incorrect outputs, on all 2^5 possible cases.
Population size:	500
Number of generations:	100
Maximum program depth:	8
Mutation probability:	0.1
Crossover probability:	0.9

Table 2: Santa Fe trail parameter tableau for CFG-GP

Raw fitness:	Number of pieces of food before the ant times out with 600 operations.
Standardised fitness:	Total number of pieces of food, minus the raw fitness.
Population size:	500
Number of generations:	100
Maximum program depth:	10
Mutation probability:	0.1
Crossover probability:	0.9

4. RESULTS AND DISCUSSION

Table 3: 6-bit multiplexer parameter tableau for CFG-GP

Raw & standardised fitness:	Number of inputs producing incorrect outputs, on all 2^6 possible cases.
Population size:	500
Number of generations:	100
Maximum program depth:	8
Mutation probability:	0.1
Crossover probability:	0.9

The results of our experiments are shown as heatmap charts in figures 1, 2 and 3. The charts show the density of programs found at each fitness/complexity point throughout all runs. A brighter cell indicates a greater density of programs, while black signals no programs.

Each of the charts illustrates particular problem traits. The boolean problems of even-five parity and 6-bit multiplexer show a higher density of programs of middle fitness or better. This should be expected since on a boolean problem, any candidate solution which attains more incorrect results than correct merely needs to undergo a mutation to negate the entire program to give a more than 50% correct program. This is in contrast with the Santa Fe trail chart where the highest density of programs is at the lowest fitness score.

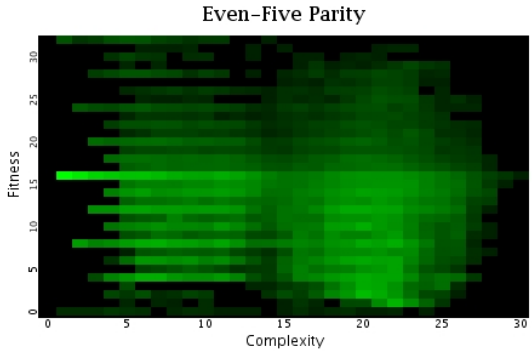

Figure 1: Density of programs discovered with each fitness/complexity score on the even-five parity problem.

It is worth noting that the results for the Santa Fe trail show that no zero fitness programs were located, so further work will be required to ensure that a suitable parameter set is being used here, but it is expected that the spread of complexities would remain similar.

The 6-bit multiplexer and Santa Fe trail both show a region of very low complexity which appears to be insufficiently complex to sustain highly fit programs. This is not the case with even-five parity, which concurs with what we know about the problem; that it is essentially easier to solve. The property that is of real interest in the context of this paper, is that the most fit programs are found with a lower density at higher complexities. This implies that it should be possible to focus evaluation efforts at lower complexities without damaging the search's ability to locate solutions.

5. FUTURE WORK

The results discussed above show that there is a potential use for the cyclomatic complexity metric in the evaluation of evolved programs. But, for it to be useful in practice it needs to be shown to hold for a wider set of problems. In particular, testing its applicability to any class of problems with an expensive evaluation

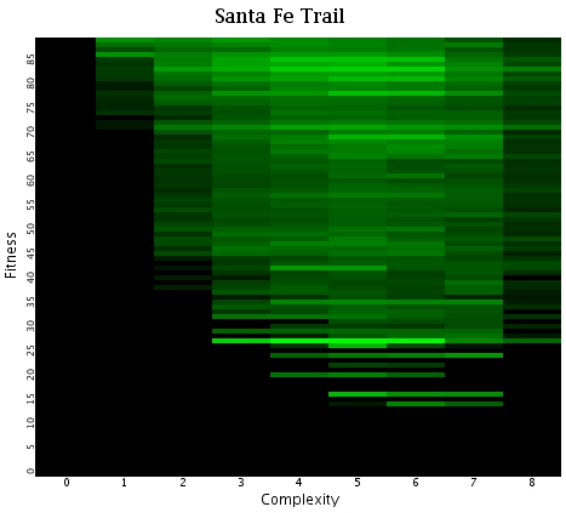

Figure 2: Density of programs discovered with each fitness/complexity score on the Santa Fe trail problem.

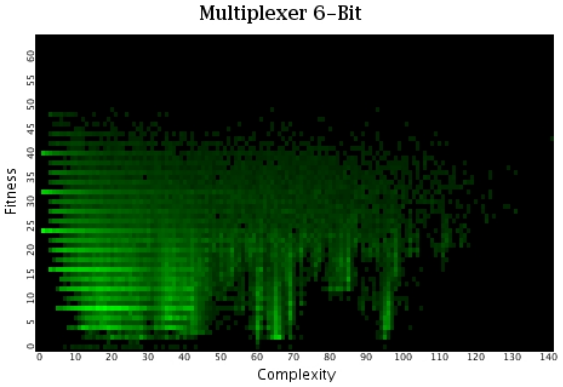

Figure 3: Density of programs discovered with each fitness/complexity score on the 6-bit multiplexer problem.

procedure, such as image processing tasks, would be valuable. It would also be necessary to analyse the specific ranges of complexities that are of interest in different problems. Beyond further analysis the next step would be the actual implementation and use of these metrics to improve performance of real runs.

An alternative benefit that has not been discussed in this paper is the use of complexity metrics to improve the readability of code by humans. This is another area for future work.

6. REFERENCES

[1] D. Andre and J. R. Koza. Parallel genetic programming on a network of transputers. In J. P. Rosca, editor, *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 111–120, Tahoe City, California, USA, 9 July 1995.

[2] T. Castle and L. Beadle. Epochx: genetic program software for research. `http://www.epochx.org/`, 2007.

[3] J. L. Elshoff. An analysis of some commercial PL/I programs. *IEEE Transactions on Software Engineering*, 2(2):113–120, 1976.

[4] A. Fukunaga, A. Stechert, and D. Mutz. A genome compiler for high performance genetic programming. In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 86–94, University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998. Morgan Kaufmann.

[5] M. H. Halstead. Natural laws controlling algorithm structure? *SIGPLAN Not.*, 7(2):19–26, 1972.

[6] S. Harding and W. Banzhaf. Fast genetic programming on GPUs. In M. Ebner, M. O'Neill, A. Ekárt, L. Vanneschi, and A. I. Esparcia-Alcázar, editors, *Proceedings of the 10th European Conference on Genetic Programming*, volume 4445 of *Lecture Notes in Computer Science*, pages 90–101, Valencia, Spain, 11 - 13 Apr. 2007. Springer.

[7] S. Kent and D. Dracopoulos. Bulk synchronous parallelisation of genetic programming. Technical Report CSTR-96-13 ; CNES-96-02, Brunel University, Uxbridge, Middlesex, UK, July 1996.

[8] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[9] T. J. McCabe. A complexity measure. In *ICSE '76: Proceedings of the 2nd international conference on Software engineering*, page 407, Los Alamitos, CA, USA, 1976. IEEE Computer Society Press.

[10] M. E. Roberts. The effectiveness of cost based subtree caching mechanisms in typed genetic programming for image segmentation. In G. R. Raidl, S. Cagnoni, J. J. R. Cardalda, D. W. Corne, J. Gottlieb, A. Guillot, E. Hart, C. G. Johnson, E. Marchiori, J.-A. Meyer, and M. Middendorf, editors, *Applications of Evolutionary Computing, EvoWorkshops2003: EvoBIO, EvoCOP, EvoIASP, EvoMUSART, EvoROB, EvoSTIM*, volume 2611 of *LNCS*, pages 444–454, University of Essex, UK, 14-16 Apr. 2003. Springer-Verlag.

[11] P. A. Whigham. Grammatically-based genetic programming. In J. P. Rosca, editor, *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 33–41, Tahoe City, California, USA, 9 July 1995.

Design of a Privacy Protected Policy-Based Authorisation System

Kaniz Fatema
Computing Laboratory
University of Kent,
Canterbury, CT2 7NZ
kf66@kent.ac.uk

David Chadwick
Computing Laboratory
University of Kent,
Canterbury, CT2 7NZ
D.W.Chadwick@kent.ac.uk

ABSTRACT
An access control system based on policy is more flexible than a hard coded id-based access control system. Recently privacy has become an important concern with regard to electronic data. Hence the necessity has arisen to include privacy policy in the access control system. These privacy policies can come from different authors such as issuer, data subject or keeper of that personal data. Organizations dealing with private data need to include these privacy policies coming from different authorities with their access control policies. In this paper a design is proposed for a system which can handle the inclusion of privacy policies from different authorities through the use of sticky policy paradigm and has capability to include multiple policy languages in the same infrastructure.

Keywords
Privacy policy, PDP, Application dependent PEP, Application Independent PEP, Master-PDP, Obligation, Obligations Service.

1. INTRODUCTION
Privacy protection of personal data is an important requirement placed upon the organisations handling electronic private data. Organisations need to collect personal data for different business, promotional, research and operational purposes. The organisations need to ensure the privacy of these data. Now-a-days people are being more concerned about the privacy of their personal data. Different laws [2, 12] also exist to support protection of privacy of personal data. To support these pressures a system designed to provide privacy of personal data should have the following capabilities:

1. **Sticky Policy Paradigm:** A mechanism to have privacy rules from different authorities such as issuer, data subject or keeper of the personal data. In a traditional access control system only the organisation's authority can set the access control policy which makes the system unsuitable for privacy protection. Our proposed system will accept policies from different authorities and will include them with the access control policy through the use of a sticky policy paradigm (where data and the related policy are stuck together).
2. **User Friendly Interface:** Need to have a user friendly interface in order to allowing the creation of privacy policy by non-technical people. We plan to provide such a user friendly interface, where users can define policy with simple tick boxes.
3. **Obligations:** Need to include obligations (for example notifying the subject or deleting the data after a certain period) with the access decision which a traditional access control system is unable to do.
4. **Distributed Enforcement:** Need to have a mechanism for distributed enforcement of privacy policy. In our proposed system the privacy policy will not only be stuck with the private data within the system or while leaving the system but also will be enforced by any receiving organisation.
5. **Multiple Policy Languages:** Need to include multiple policy languages. In a distributed environment it cannot always be assumed that all the systems will use the same policy language. If a privacy policy with the private data arrives at a system which does not support that particular policy language then the system will not be able to implement the policy. Our proposed system will co-ordinate many different policy languages. Conflict resolution strategies are also designed to resolve conflicts among decisions returned by many different Policy Decision Points (PDPs).

Section 2 includes a literature review of the current research in this field. A brief description of the system is provided in section 3; conclusions and acknowledgements are covered in section 4 and 5.

2. REVIEW OF RELATED RESEARCH
IBM's security research group has been doing research on privacy protection of customer's data collected by enterprises [4-6, 1, 10]. They used sticky policy paradigm where personal data are associated to the privacy policy and passed together while exchanging data among enterprises. This research on privacy does not provide a way to accommodate different policy languages. The obligations of obligation model are simply activity names such as log, notify, getConsent etc. without a way to actually enforce the obligation [6]; where our model provides a way to implement the application independent obligations.

Marco Casassa Mont of HP has provided a way of transmitting encrypted confidential data with obligations to other parties[8], [9]. They used the sticky policy as an IBE (Identity Based Encryption) key for obfuscation of data [9]. The trusted authority will issue the decryption key only if the requester acknowledges the compliance to the disclosure policies. Nevertheless, this model does not include multiple policy languages and the trusted authority could be considered a single point of failure. Our model ensures the distributed enforcement of sticky policy through the use of obligation.

3. ARCHITECTURE OF THE SYSTEM

In our model the conventional PEP (Policy Enforcement Point) of XACML [11] is replaced by two components – **AppDep PEP**, which works as a conventional PEP and **App**lication **Ind**ependent **PEP** which acts as an interface to the application and helps to implement application independent obligations.

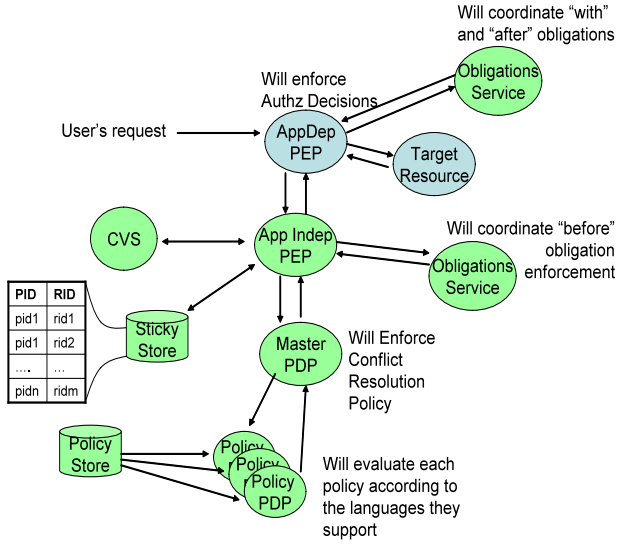

Figure 1. Privacy protected authorisation model

Obligations are actions that must be performed when a certain event occurs. When the event is an authorisation decision, then the obligations are actions that must be performed before, after or along with the enforcement of the authorisation decision. **Obligations Service** is the component responsible for enforcing these obligations which we distinguished as *before, after* and *with* obligations.

The **CVS** (Credential Validation Service) does the validation of credentials issued by different authorities.

The **Policy Store** stores each policy with a given unique ID - the Policy ID (PID). The **Sticky Store** keeps binding of PIDs to Resource ID (RID). The **Master PDP** is the component that calls multiple subordinate PDPs (Policy Decision Point) and combines the decisions returned by the PDPs to form a single access decision. The Master PDP has a **conflict resolution policy** that resolves the conflict among the authorisation decisions returned from the multiple subordinate PDPs

Each **subordinate PDP** is comprised of the policies. When an access request comes to a PDP it consults the policies and returns a decision and optionally obligations. Each PDP has the following attributes:

- the PID of the policy it is loaded with
- the author of the policy (used to determine the policy authority)
- the date that the policy was written
- the list of resource types covered by the policy (used to determine its specificity)

3.1 Conflict Resolution Policy

A conflict resolution policy (CRP) consists of multiple conflict resolution rules (CRRs). The default CRP is read in at program initialisation time and additional CRRs obtained from the subjects' and issuers' sticky policies can be dynamically added to it. Each conflict resolution rule (CRR) comprises a condition, a decision combining rule (DCR), an author and a time of creation. The conditions of the CRRs are tested against the request context by the Master PDP to see which decision combining rule to use.

All the PDPs are ordered according to a built-in precedence rule. When the DCR of the chosen CRR is *first applicable* the Master PDP calls each subordinate PDP in order, and stops processing when the first grant or deny decision is obtained. Any *not applicable* results are ignored whilst any *indeterminate* results are only returned to the AIPEP if there is a problem with the request context such as format mismatch.

For *deny overrides* and *permit overrides* the Master PDP will call all the subordinate PDPs and will combine the decisions with the following semantics:

- DenyOverrides – A Deny result overrides all other results. If there is no Deny result, then Indeterminate overrides Permit and NotApplicable. If there is no Indeterminate or Deny result, Permit overrides NotApplicable.
- PermitOverrides – A Permit result overrides all other results. If there is no Permit result, then Indeterminate overrides Deny and NotApplicable. If there is no Indeterminate and Permit result then Deny overrides NotApplicable.

3.1.1 Precedence Rules

The built-in precedence rules determine the precedence to resolve conflicts of PDP and CRR.

The following precedence rules are used, in order:

1. The higher authority overrides lower authority [7]. Higher authority is determined from the authority hierarchy. The highest authority is the Law, and then the issuer of the data, then the data subject and finally the keeper of the data has the least precedence.
2. If more than one policy or CRR is written by the authority at the same level of hierarchy, then the specific overrides the general [3],[7], [13] precedence is used.
3. If more than one policy is available for the same specific resource/subject, then the new overrides the old [7] precedence is used. This is determined from the date the policy was written.

4. CONCLUSION

We have designed the system in a way that when implemented organisations can integrate it with a minimum alteration to their existing systems and the systems can do the privacy enforcement automatically.

ACKNOWLEDGEMENTS

Thanks to the TAS3 project for funding this work.

5. REFERENCES

[1] Berghe, C. V., Schunter, M. 2006. Privacy Injector — Automated Privacy Enforcement through Aspects. 6th Workshop on Privacy Enhancing Technologies Robinson College, Cambridge, United Kingdom.

[2] Data protection Act 1998, available on http://www.opsi.gov.uk/Acts/Acts1998/ukpga_19980029_en_1

[3] Dunlop N., Indulska, J., Raymond, K.2003. Methods for Conflict Resolution in Policy-Based Management Systems, Proceedings of Seventh International Enterprise Distributed Object Computing Conference, 16-19 Sept. 2003.

[4] Karjoth G., Schunter, M. 2002. A Privacy Policy Model for Enterprises, 15th IEEE Computer Foundations Workshop, June 24-26, 2002.

[5] Karjoth G., Schunter, M., Waidner, M. 2002. Platform for Enterprise Privacy Practices: Privacy-enabled Management of Customer Data. 2nd Workshop on Privacy Enhancing Technologies (PET 2002), San Francisco, CA, USA. April 14-15, 2002.

[6] Karjoth, G., Schunter, M., Waidner, M. 2002. Privacy-enabled services for enterprises. Proceedings of the 13th International Workshop on Database and Expert Systems Applications, 2-6 Sept. 2002.

[7] Linington, P., Milosevic, Z., Raymond, K. 1998. Policies in Communities: Extending the ODP Enterprise Viewpoint. Proceedings of the second International Enterprise DISTRIBUTED Object Computing Workshop (EDOC '98), La Jolla, California, USA, Nov 1998.

[8] Mont M. C., Pearson S., Bramhall P. 2003. Towards Accountable Management of Identity and Privacy: Sticky Policy and Privacy. Trusted System Laboratory, HP Laboratories, Bristol, HPL-2003-49.

[9] Mont, M. C. 2004. Dealing with Privacy Obligations: Important Aspects and Technical Approaches. International conference on trust and privacy in digital business No1, Zaragoza.

[10] Nelson, M. R., Schunter, M., McCullough, M.R., Bliss, J.S. 2005. Trust on Demand — Enabling Privacy, Security, Transparency, and Accountability in Distributed Systems. 33rd Research Conference on Communication, Information and Internet Policy (TPRC), September 23-25, Arlington VA, USA.

[11] OASIS .eXtensible Access Control Markup Language (XACML) Version 2.0. OASIS Standard, 1 Feb 2005.

[12] OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data, available on http://www.oecd.org/document/18/0,3343,en_2649_34255_1815186_1_1_1_1,00.html

[13] Russello, G., Dong, C., Dulay, N. 2007. Authorisation and conflict resolution for Hierarchical Domains. Eight IEEE International Workshop on Policies for Distributed systems and Networks (POLICY '07).

Force-Directed Layout for Euler Diagrams

Luana Micallef
University of Kent
Canterbury, UK
lm304@kent.ac.uk

Peter Rodgers
University of Kent
Canterbury, UK
P.J.Rodgers@kent.ac.uk

ABSTRACT
Euler diagrams are the only diagrams that intuitively represent containment, intersection and exclusion of data, but none of the current automatic diagram layout techniques produce desirable layouts in a reasonable time. We adopt a force-directed approach to automatically lay out aesthetically pleasing Euler diagrams in a relatively fast time. A Java prototype demonstrates our novel force model.

1. INTRODUCTION
Euler diagrams can intuitively represent containment, intersection, exclusion and other relationships between data sets. Consequently, these diagrams are used in a wide variety of application areas, such as biological visualization [5], and classification and querying of large databases [7]. As illustrated in Figure 1, these diagrams are composed of simple closed curves. As these curves meet, the plane is split up into zones that are uniquely identified by the curves that contain them.

Producing an aesthetically pleasing Euler diagram layout is a difficult task. Recently, approaches such as evolutionary optimization [5], hill climbing and aesthetic layout metrics [4] were adopted to automatically lay out these diagrams. Such methods are computationally expensive and thus, they are impractical for laying out diagrams with a large number of curves.

We adapt the well-known and successful force-directed approach used in graph drawing [2] to lay out Euler diagrams. For graphs, a system of springs (the edges) and electrically charged particles (the vertices) forms a force model, which is brought to stability in a sequence of iterations. Noting that closed curves in an Euler diagram can be represented as polygons, forces between vertices, lines, and the polygons themselves are used to produce nicely laid out diagrams. A major challenge, which is not a problem in graph layout, is the development of an appropriate force model that maintains the original structure of the diagram. By this we mean that all the zones that are in the original diagram are still in the laid out diagram, and that no new zones are added.

Force-directed placement has not previously been used to lay out Euler diagrams, but from the ongoing research described here, we have learnt that good layouts can be produced from previously generated diagrams in a reasonable time.

2. OUR APPROACH
Automatic generation methods for Euler diagrams such as [3, 6] produce difficult to comprehend representations (see Figure 1, left column). Thus, after generation, these representations have to be refined and improved by other specific layout techniques.

In our case, our initial diagrams are generated by a technique based on that given in [6]. These diagrams are wellformed and thus, they have no poor structural features such as triple points, concurrency or disconnected zones [3]. As a result, the technique in [6] does not generate all possible diagrams.

Given an initial diagram, our approach simulates the effect of a group of attractive and repulsive forces for a number of iterations. The polygons are changed and the diagram is improved until finally, stability is reached. The main objectives of the forces are to: i) attain regular, smooth and similarly shaped polygons; ii) maintain the original structure of the diagram; and iii) ensure adequately sized polygons and zones.

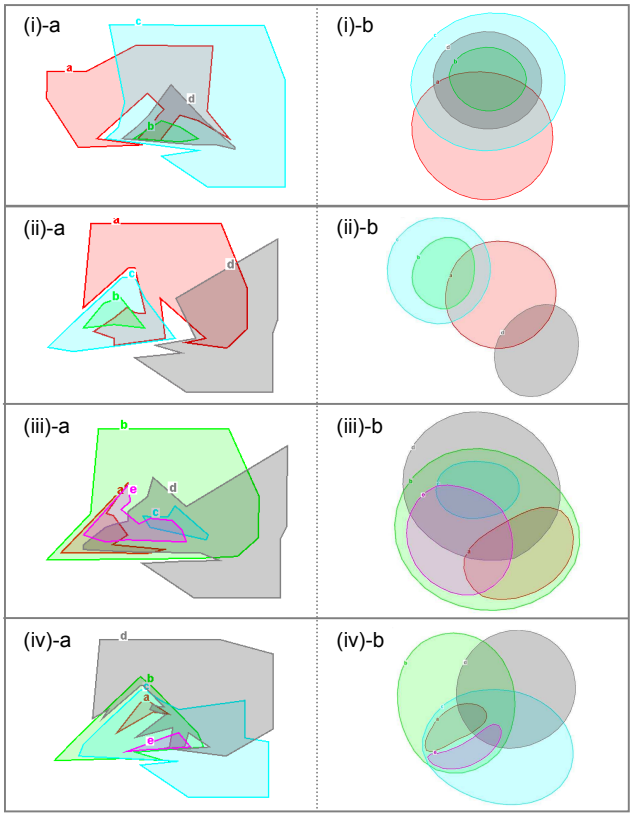

Figure 1. Initial 4 and 5-curve diagrams (left column - a) and their final layouts produced by our force-directed approach (right column - b).

2.1 Forces

Similarly to graph-based force-directed approaches, two types of forces are used in our model: attractive and repulsive forces. Attractive forces are directly proportional to the distance between two vertices, and repulsive forces are inversely proportional to the squared distance between two vertices.

Two forces attempt to attain smooth and regular polygons: **(a)** a repulsive force between every pair of vertices of the same polygon; and **(b)** an attractive force between every pair of adjacent vertices of the same polygon. Thus, as all the vertices of a polygon move away from each other, the lines (acting as springs) attract neighbouring vertices so that a smooth polygon is formed.

However, since regular polygons might not always be appropriate to represent the required zones as the structure of the diagram must be maintained, a set of other forces that handle all the possible polygon-to-polygon relationships are required. Thus, if in the original diagram two polygons *do not intersect*, **(c)** vertices of the two different polygons repulse away from each other. If in the current layout these polygons intersect, **(d)** vertices (of the different polygons) that are currently breaking the structure attract. If in the original diagram polygons *intersect*, **(e)** vertices (of the different polygons) that are currently in the common zone repulse away from each other. This helps to ensure the existence of the zone. If in the current layout these polygons do not intersect, a special attractive force, **(f)** which is inversely proportional to the squared distance between vertices of the first and second polygon, is necessary to recover the lost zone. If in the original diagram, a polygon *contains* another polygon, **(g)** vertices of the two polygons repulse away from each other. If in the current diagram, the first polygon does not contain the second, **(h)** vertices (of the different polygons) that are invalidating the structure attract. Since there might be cases where a vertex of one polygon is closest to a point on a line between two vertices of a second polygon, the above mentioned forces are applied between the vertex of the first polygon and the closest point on the line of the second polygon. In this way, the system helps to ensure that none of the moving vertices invalidate the diagram structure.

The final set of forces are specifically designed to ensure adequately sized polygons and zones. Noting that the comprehension of diagrams consisting of various n-curve zones (n refers to the number of polygons in which the zone lies) can be improved by increasing the size of outer zones (which have a small n) and decreasing the size of the inner zones (which have a big n), the ideal size for every n-curve zone is calculated by $2^m/2^n$ (where m is the number of curves in the diagram). To increase the size of the zone, the size of the containing polygon is increased and the polygon is moved. To increase the size of the polygon, the repulsive force that handles its smoothness - force (a) - is progressively increased. To increase the actual zone size, **(i)** the polygons containing the zone are moved by an attractive force between their centroid and the zone centroid. If a new, unwanted, zone is introduced, **(j)** the polygons containing the new zone are moved away by a repulsive force between their centroid and the zone centroid. If alternatively a wanted zone is lost, **(k)** the polygons that should contain the zone are pushed towards each other by an attractive force between their centroids. Such corrective forces are essential to correct any flaws in the structure during the layout process. Still related to diagram aesthetics, **(l)** contained polygons are attracted to the centroid of the containing polygon or zone to centre internal curves.

3. RESULTS

We implemented a Java prototype for our force-directed layout approach and we tested it using various 3, 4 and 5-curve wellformed diagrams generated by [6]. Figure 1 illustrates some before and after layouts. The final layout for all the 9 3-curve and all the 114 4-curve diagrams maintained the original description and we judged them to have acceptable layout. On average, 3-curve diagrams were laid out in 7 seconds[1] and 4-curve diagrams were laid out in 26 seconds. For 5-curve diagrams, 208 out of 342 final layouts successfully maintained the original description and by our judgement they had an acceptable layout. On average, 5-curve diagrams were laid out in 77 seconds.

4. CONCLUSIONS AND FUTURE WORK

We have developed a novel force model to produce aesthetically pleasing and comprehensible 3, 4 and 5-curve Euler diagram layouts, relatively quickly. This is still ongoing research and hence, refinements to the force model and algorithm are likely to achieve further aesthetic and performance improvements.

A main drawback of this approach is that for some cases, particularly dense diagrams, the interacting forces become unmanageable and thus, it is difficult to design a force model that handles all possible diagrams. However, results indicate that it works for all 3, 4-curve and most 5-curve diagrams.

An issue with this work is the evaluation of the quality of the final layouts. Various aesthetic metrics have been developed for graph drawing, but few are available for Euler diagrams. Only one empirical study has been carried out [1] and from this, the need for criteria to avoid polygon jaggedness, inequality of zone areas and curve closeness, has been identified. Although these criteria are dealt with by our force model, further evaluation based on empirical studies and general information visualization and perceptual principles [8], needs to be conducted. It is possible to compare layouts (i)-b and (ii)-b in Figure 1 (produced by our approach) with the corresponding layouts (i) and (ii) in Figure 2 (produced by [4]). We believe that the layouts produced by our force model are more aesthetically pleasing and comprehensible.

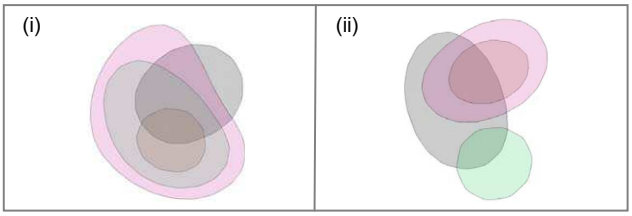

Figure 2. Final layouts produced by [4].

Currently, only wellformed diagrams are handled. Since generation algorithms can produce diagrams that are not wellformed, current forces in our model could be modified and new ones could be added, to ensure the successful layout of such diagrams.

The code is still experimental and efficiency is currently not a primary concern. Clearly, many efficiency gains can be made by general code speedups and by applying known optimization methods used in other force-directed approaches.

[1] An Intel Core 2 Duo CPU E7200 @ 2.53GHz with 4GB RAM running Microsoft 32-bit Windows XP Professional (version 2002, Service Pack 3) and Java VM version 1.6.0_18-b07 was used.

5. REFERENCES

[1] F. Benoy and P. Rodgers. Evaluating the Comprehension of Euler Diagrams. *Proc. IV 2007*, pages 771–778. IEEE.

[2] P. Eades. A Heuristic for Graph Drawing. *Congressus Numerantium*, 42:149–160, 1984.

[3] J. Flower and J. Howse. Generating Euler Diagrams. *Proc. Diagrams 2002, LNCS 2317*, pages 61–75. Springer.

[4] J. Flower, P. Rodgers, and P. Mutton. Layout Metrics for Euler Diagrams. *Proc. IV 2003*, pages 272–280. IEEE.

[5] H. A. Kestler, et al. VennMaster: Area-proportional Euler diagrams for functional GO analysis of microarrays. *BMC Bioinformatics*, 9(1), 2008.

[6] P. Rodgers, L. Zhang, G. Stapleton, and A. Fish. Embedding Wellformed Euler Diagrams. *Proc. IV 2008*, pages 585–593. IEEE.

[7] A. Verroust and M.-L. Viaud. Ensuring the Drawability of Extended Euler Diagrams for up to 8 Sets. *Diagrams 2004, LNCS 2980*, pp. 128–141. Springer.

[8] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

Dynamic Generation of Adaptation Plans for Self-adaptive Software Systems

Carlos de Silva
Computing Laboratory
University of Kent
Canterbury, UK, CT2 7NZ
ces26 @ kent.ac.uk

ABSTRACT

The generation of adaptation plans, one of the activities of a self-adaptive software system, is a complex process that depends on several factors, which may change during the system operational lifetime, such as its operational state and environment. Hence, self-adaptive software systems should be able to generate adaptation plans during run-time. In this direction, this paper presents a general overview of our research for developing a framework for the automatic generation of adaptation plans based on the use of system models, artificial intelligence planning techniques and workflows. In order to evaluate the proposed approach, we have conducted some experiments where this framework is used for generating during run-time the workflows that coordinate the architectural reconfiguration of a self-adaptive software system.

1. INTRODUCTION

It is commonly agreed that a self-adaptive software system should be able to modify its own structure and/or behaviour at run-time due to changes in the system, its requirements, or the environment in which it is deployed. In order to determine the actions to be taken to adapt itself, a self-adaptive software system observes and analyses itself and its environment, and if an adaptation is deemed to be necessary, an adaptation plan is generated for altering the system in a controlled manner. These activities are usually captured in terms of a feedback control loop containing four key phases, that is, monitoring, analysis, planning and execution.

The process associated with the self-adaptation of software depends on several factors that may change during the system operational lifetime. Hence, adaptation plans should be dynamically generated for dealing with the variability and uncertainty involved in the self-adaptation process. Some existing approaches for self-adaptation focus on the definition of mechanisms for the selection of adaptation plans using policy-based languages for the specification of adaptation plans [2, 3], while others have focused on the infrastructure for executing adaptation plans [6, 7]. These approaches require the definition at design-time of each possible adaptation plan and each system condition that triggers an adaptation. However, at design-time it is not possible to anticipate all possible contexts of self-adaptation. For example, when dealing with architectural reconfiguration of web services, one is not able to know at design-time, all available resources and possible configurations from which the reconfiguration plans are defined.

This problem has been partially solved by using Artificial Intelligence (AI) planning technology to generate adaptation plans together with the system architectural configuration [1]. However, by mixing the selection of a configuration with the generation of plans, the scalability of the planning task has been affected. Moreover, this approach also requires the definition at design-time of a different planning model for each application being targeted. Another solution that has looked into the same problem has explored techniques for comparing models and priority rules for identifying adaptation plans, which allows the reconfiguration of a software architecture by promoting its reuse in different applications and execution platforms [4]. However, this approach also requires the definition at design-time of all possible variations in the system architecture, and all possible adaptation rules. Moreover, it has been shown that the use of priority rules is not enough for generating plans [5] (e.g., dealing with the reconfiguration of components in arbitrary states and complex dependencies relationships among them).

In this context, the aim of this research is to investigate the automatic generation of adaptation plans for self-adaptive software systems. Our objective is to develop a framework for dynamically generating adaptation plans based on the use of model-based, AI planning and workflow management technologies. We want to provide a dynamic solution that does not need to know at design-time all the possible architectural configurations for the system, and that improves the scalability of the process for generating adaptation plans.

2. SOLUTION OVERVIEW

We follow the trend in which the generation of adaptation plans is divided in two problems: i) the selection of a configuration; and ii) the generation of the workflow for instantiating the selected configuration. In this context, our work is focused on the problem of generating a reconfiguration plan (a workflow) for a given configuration, assuming the existence of mechanisms responsible for selecting a configuration for the system.

Our approach for the generation of reconfiguration plans is further divided in two levels of abstraction: strategy and tactics. At the strategy level, we generate abstract workflows considering the workflow objective and its structure in terms of its constituent tasks and the control flow relationship among them, without identifying the actual resources that will be used during the plan execution. At the tactics level, we generate concrete workflows by identifying the actual resources that will be used during the plan execution. It is important to note that at the strategy level, the resources associated with the tasks are referred to by a logical name, which should be sufficient to identify the actual resources at the tactics level. In this way, an abstract workflow can be implemented using different combinations of resources.

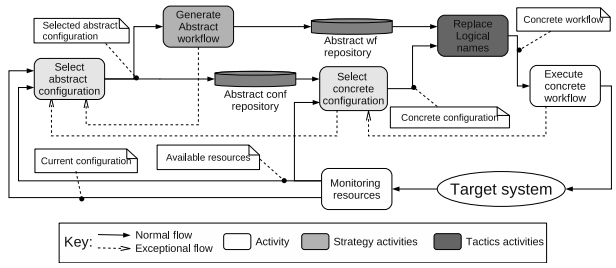

Figure 1: Overview of the workflow generation process.

Following the division of reconfiguration plans in strategy and tactics, configuration models are also divided in two levels of abstractions. Hence, an abstract configuration model describes a system configuration in terms of the functional properties of its architectural elements, identifying the structure of the system, but abstracting away from the actual instances of the architectural elements. On the other hand, a concrete configuration model describes a system configuration in terms of the actual architectural instances, and their respective attributes. Therefore, in a manner similar to plans, an abstract architectural configuration can be instantiated into different concrete configurations depending on the availability of actual architectural instances.

An overview of the generation process is presented in Figure 1. At the strategy level, we obtain the current system configuration and the selected abstract configuration. The configuration models are translated into a planning problem that is used as input to an AI planner. The planner output is then translated into a workflow specification. The activities of this level are represented by the Generated abstract workflow activity. At the tactics level, an abstract workflow is converted into a concrete workflow by replacing the logical names with the actual resources identified in the selected concrete configuration. If there is a problem while executing a concrete workflow, a new concrete configuration is selected and a new concrete workflow is generated. If it is not possible to generate a concrete workflow (e.g., there are not enough resources), the process goes back to the strategy level, where a new abstract workflow is generated. This division between strategy and tactics helps to reduce the time necessary for finding a plan, since it is not necessary to know about all available resources in the environment, but only the resources involved, represented using logical names.

3. PRELIMINARY RESULTS

In order to evaluate our work, we have developed a prototype infrastructure for supporting the generation process, and used this infrastructure for conducting some experiments. We have also developed a second generation process based on the work presented in [1], where the system configuration is selected by an AI planner. These experiments considered the time necessary for generating a concrete workflow for deploying a configuration, where we changed the number of components and connections involved in the configuration for changing the size of the generated workflow. Our initial results comparing both processes is presented in Figure 2.

Our initial results demonstrate the feasibility of our approach for generating adaptation plans. However, more experiments are necessary to fully evaluate our approach, including configurations with bigger number of components for evaluating its scalability, the use of other planning techniques, and further comparison with other existing approaches.

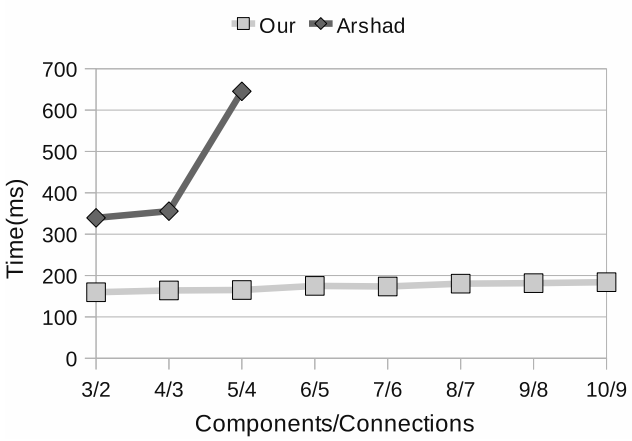

Figure 2: Initial experimental results.

4. CONCLUSIONS AND FUTURE WORK

The main focus of our work is the definition of a framework for automatic generation of workflows. In this paper, we have presented a general overview of our generation framework and its application in the context of self-adaptive software systems. Our next steps include the investigation of exception handling mechanisms for dealing with failures during the execution of dynamically generated workflows, and the application of the defined generation process in the context of self-adaptation of business processes.

5. REFERENCES

[1] N. Arshad et al. Deplotment and dynamic reconfiguration planning for distributed software systems. *Software Quality J.*, 15(3):265–281, 2007.

[2] S. W. Cheng, D. Garlan, and B. Schmerl. Architecture-based self-adaptation in the presence of multiple objectives. In *Proc. of ICSE SEAMS'06*, pages 2–8, 2007.

[3] J. C. Georgas and R. N. Taylor. Towards a knowledge-based approach to architectural adaptation management. In *Proc. of the WOSS'04*, pages 59–63, 2004.

[4] B. Morin et al. An aspect-oriented and model-driven approach for managing dynamic variability. In *Proc. of the MoDELS'08*, volume 5301 of *LNCS*, pages 782–796, 2008.

[5] C. Shankar and R. Campbell. Ordering management actions in pervasive systems using specification-enhanced policies. In *Proc. of PERCOM'06*, pages 234–238, 2006.

[6] S. Shrivastava et al. A workflow and agent based platform for service provisioning. In *Proc. of EDOC'00*, pages 38–47, 2000.

[7] G. Valetto and G. Kaiser. Using process technology to control and coordinate software adaptation. In *Proc. of ICSE'03*, pages 262–272, 2003.

Provenance-Awareness in R

C.A.Silles and A.R.Runnalls
Computing Laboratory
University of Kent
Canterbury, UK, CT2 7NZ
{C.A.Silles,A.R.Runnalls}@kent.ac.uk

ABSTRACT

The use of computer systems for recording information has proliferated in recent years; however, facilities for recording *how* this data has come to be in its present state have only recently started to catch up. The goal of the *provenance-aware computing* field is to enable computer systems to record the *provenance* — a record of *lineage* or *pedigree* — of data in such a way that enables users to query and effectively use this previously unrecorded information. In this paper we look at how facilities for recording and retrieving provenance have been introduced to the interactive statistical environment and programming language CXXR, which is a C++ flavour of the popular R project.

1. INTRODUCTION

The term *provenance* has — according to the Oxford English Dictionary — been in use since the 18th century to mean "The fact of coming from some particular source or quarter; origin, derivation", and since the 19th century to refer to "The history of the ownership of a work of art or an antique, used as a guide to authenticity or quality; a documented record of this". Today, provenance is a well-understood concept in many different areas, including art, antiques and memorabilia; however, it is only relatively recently that the term has been used in the context of computing.

1.1 Provenance-Aware Computing

We consider the *provenance* of an item of data to be "the process that led to that piece of data" [5]. Information that describes the origin of data is becoming not only of increasing use, but also necessity, as computer systems have taken on significant roles in many disciplines recording and managing data. There is currently significant interest in creating provenance-aware computer systems for use in areas as diverse as e-science, medical physics (CT, MRI, fMRI, PET etc.), proteomics, finance and weather monitoring. This field has developed rapidly over the last decade, and is now reaching maturity with the Open Provenance Model for the representation and exhange of provenance information [7].

1.2 CXXR

CXXR is a variant of R, which is an open-source implementation of S. S is a language and interactive environment for statistical computing, graphics and exploratory data analysis [1]. It was developed during the mid-1970s at Bell Labs by John Chambers and Richard Becker. S emerged from Bell Labs at around the same time as the C programming language, and this is reflected in both its syntax and name. Despite this, S uses the semantics of a functional programming language, including employment of lazy evaluation.

The most significant landmark in the history of S was reached in 1988 when 'New S' was released in 1988, sporting a new feature entitled *S AUDIT* [2]. While a user operated a session within New S, a record was maintained of each top-level expression evaluated, as well as objects read from and written to during the course of evaluation. The accompanying S AUDIT program was able to process this record and allow the user discover details of the session, including the full sequence of statements evaluated; which statements are responsible for reading from, or writing to, a particular object; or simply providing a list of all objects in the session. Therefore, New S became one of the first provenance-aware software applications, and even featured a primitive provenance visualisation in the shape of an *audit plot*: features that were at the time innovative, and remain novel to this day.

While S as an application continues life as a commercial product called S+ retailed by TIBCO [10], the language, library and environment have been reimplemented as part of the open-source R project [9]. Crucially, R has never had S AUDIT-like capabilities.

CXXR is a project to reengineer the fundamental components of the R interpreter from C into C++ while fully preserving functionality of the standard R distribution [8].

Within R — and CXXR likewise — the user evaluates expressions on a command line. An expression entered on the command line is referred to as a *top-level expression*, as opposed to a subexpression such as `1+2` in the top-level expression `a <- 1+2`. When this top-level expression is evaluated, a *binding* is created between the *symbol* `a` and the *object* that results from evaluating the subexpression `1+2` — an *integer vector* comprising a single element, 3.

1.3 Provenance-Aware CXXR

The principal objective of this work is to enable CXXR to identify the following information of a given binding: -

- The **process** that led to it – the sequence of commands executed;
- Its **ancestors** – which other bindings it depends on;
- Its **descendants** – which other bindings depend on it.

Due to the divorce of object values from the symbols by which they are referenced, a novel approach to provenance attribution is required. Rather than recording reads from and writes to *objects*, we need to record reads from and writes to *bindings*.

2. IMPLEMENTATION

The fundamental addition to CXXR required for recording provenance is the introduction of read and write *monitors*, which are triggered when a binding is either read from or is created or overwritten.

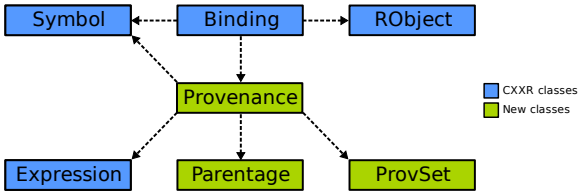

Figure 1: Class collaboration diagram

2.1 Storing

Three C++ containers have been introduced to store various aspects of provenance information.

The `Provenance` class is central to storing provenance for a binding. It is composed of the *timestamp* of when the binding was created; the top-level *expression* that was being evaluated; the *symbol* that is bound; and references to the *parentage* and *children* of the binding.

Binding B1 is a **parent** of binding B2 (and conversely B2 is a **child** of B1) if binding B1 was read in the course of evaluating the top-level expression that gave rise to binding B2. Parentage is represented by the `Parentage` class, which inherits from the C++ Standard Template Library (STL) `std::vector` class, and stores pointers to `Provenance` objects.

A `ProvSet` of provenance objects is used to store references to `Provenance` objects. This collection is an `std::set`, and its members are ordered by time of creation. It is used primarily for storing references to children.

The class collaboration diagram for the relationship between new classes and existing CXXR classes is shown in Fig 1.

2.2 Recording

The mechanism responsible for reading commands from the standard input, evaluating them, and printing the result is known as the Read-Evaluate-Print-Loop (REPL). Provenance for each REPL iteration is recorded according to the following algorithm: -

- Begin with the following empty collections:
  - *Seen* set: Provenance of bindings either read from or written to;
  - *Parentage* list: Provenance of bindings read from (in sequence).
- On read of binding to symbol `x`:
  - If `x` is not in the *Seen* set, add it to *Parentage* and *Seen*.
- On write of binding to symbol `y`:
  - Create a new *Provenance* object comprising:
    - A reference to the current top-level expression;
    - A reference to symbol `y`;
    - A reference to the current *Parentage*;
    - The current timestamp;
    - An empty set of children;
  - Register the new Provenance object as a child of each of its parents, as recorded by the current Parentage list;
  - Associate this *Provenance* object with the *Binding* of `y`;
  - Add `y` to *Seen*.

2.3 Retrieval

In order for the user to be able to interrogate provenance information a couple of new R commands have been introduced. The `provenance(x)` function returns a list detailing the provenance of the current binding of `x`: the date and time of its creation, the expression immediately responsible for its current state, its symbol, and a list of both its parent and child Provenances.

The `pedigree(x)` function describes the full sequence of commands executed that led to the current binding of `x`. A full ancestry is collated by recursively looking at each Provenance's parentage starting from `x`; ordering all ancestors by time of binding creation; and printing their respective expressions, which are by definition relevant and their order chronological.

3. CONCLUSION

This work demonstrates how it is possible to introduce facilities for provenance-awareness into an interactive, command-line driven statistical environment.

Recording process documentation for the purpose of reproducible computing in R has previous been researched in *Sweave* [4], a system based on concepts of literate programming [6]. Making applications provenance-aware is not in itself a new concept [3]; however, CXXR presents some unique challenges. These include the way in which provenance is represented conceptually as an attribute of a binding is novel; the user interface concerns; as well as how individual language features are necessarily modelled to capture complete provenance.

Looking forward, one of our foremost priorities is to enable *cross-session provenance tracking*. That is to say, when the user terminates a session, the objects are serialised along with relevant provenance information so the user is able to restore the session with not only object data, but also the provenance of how the data was derived.

CXXR is currently only provenance-aware in two areas: the user workspace, and the standard library. Provenance-awareness will eventually be extended to cover all other areas, such as local functions, and new methods for inspecting provenance will need to be designed.

4. REFERENCES

[1] R. A. Becker. A brief history of S. *Computational Statistics – Papers Collected on the Occasion of the 25th Conference on Statistical Computing at Schloss Reisensburg*, pages 81–110, 1994.

[2] R. A. Becker and J. M. Chambers. Auditing of Data Analyses. *SIAM Journal on Scientific and Statistical Computing*, 8:747–760, 1988.

[3] S. P. Callahan, J. Freire, C. E. Scheidegger, C. T. Silva, and H. T. Vo. Towards provenance-enabling paraview. pages 120–127, 2008.

[4] R. Gentleman. Reproducible research: A bioinformatics case study. *Statistical Applications in Genetics and Molecular Biology*, 4(1):Article 2, 2005.

[5] P. Groth, S. Miles, V. Tan, and L. Moreau. Architecture for provenance systems, October 2005.

[6] D. E. Knuth. Literate programming. *Comput. J.*, 27(2):97–111, 1984.

[7] L. Moreau, B. Clifford, J. Freire, Y. Gil, P. Groth, J. Futrelle, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, Y. Simmhan, E. Stephan, and J. V. den Bussche. The open provenance model — core specification (v1.1). *Future Generation Computer Systems*, December 2009.

[8] A. R. Runnalls. CXXR project. `http://www.cs.kent.ac.uk/projects/cxxr`.

[9] The R Foundation. The R Project for Statistical Computing. `http://www.r-project.org`.

[10] TIBCO Software Inc. Spotfire S+. `http://spotfire.tibco.com`.