Reasoning with Diagrams: Track Record

This is a collaborative proposal between the Universities of Brighton and Kent. The team is ideally positioned to carry out the proposed programme of research, offering a combination of skills in diagrammatic reasoning, visual modelling, logic and automated reasoning, and tools for diagram manipulation and visualisation. The principal investigators have worked together for six years and have collaborated on the EPSRC projects *Formal Underpinnings of Object Technology* (GR/K67304, 1995-1999) and *Precise Visual Patterns for the Evolutionary Migration of Legacy Systems to Reusable Components* (GR/M02606, 1998-2001). *Formal Underpinnings* investigated the conceptual basis of object-oriented modelling notations and initial steps were made towards the formal semantics for various aspects of UML. The project received an *Alpha 4* grade. *Precise Visual Patterns* is an ongoing collaboration with EDP plc and applies visual modelling notations to the problem of legacy system migration.

University of Brighton, School of Computing and Mathematical Sciences (UB)

The team at the University of Brighton are all members of the visual modelling research group in the School of Computing and Mathematical Sciences. The *principal investigator* at this site, **Dr John Howse** is Reader in Mathematics and leads the group. His research focuses on the development, formalisation and application, particularly in object-oriented software development, of precise visual modelling notations. He has taught courses to industry and presented conference tutorials in visual modelling.

Howse is supported by **Dr John Taylor** and **Dr Jean Flower**. **Taylor** is Head of the School and a topologist. For the last three years, Howse and Taylor have collaborated in the field of diagrammatic reasoning, extending the work of, among others, Shin [67] and Hammer [51], in formalising and developing sound and complete inference systems for diagrammatic notations [16,23,24,25,26,27]. They have also worked with Shin on ontological relationships between abstract and concrete syntax in diagrammatic systems [22]. One of the aims of this project is to extend these results to object-oriented constraint languages. **Flower** is an algebraic topologist by training. She was appointed to a Senior Lecturer post after obtaining an MSc with distinction in Software Engineering from UB. Her MSc project [9] focussed on the task of creating a concrete representation of any given abstract diagram. It used concepts from topology and graph theory to build a new algorithm for creating concrete diagrams. The models were implemented in Java. This work has been extended in collaboration with Howse [10].

The UB *named researcher* is **Fernando Molina**. His PhD thesis [33] has recently been accepted, subject to minor amendments. He worked, under the supervision of Howse and Taylor, on developing sound and complete diagrammatic reasoning systems for extended Venn-Peirce systems, and already has five refereed publications [23,24,25,26,27].

University of Kent, Computing Laboratory (UKC)

The team at the University of Kent are drawn from two research groups in the Computing Laboratory: software and systems engineering and theoretical computer science. The *principal investigator* at this site, **Dr Stuart Kent** is a Senior Lecturer and leads the software and systems engineering group. His current research focuses on the development and application of precise, visual modelling notations in systems development [28,29,19,31,32]. Much of this work has been supported by the EPSRC [GR/M02606]. He is the inventor of constraint diagrams [28] and, with Howse, constraint trees [31]. Kent is a recognised expert in the UML. He is closely involved with the current effort to revise UML to version 2.0. This has been informed by earlier work [7,5], and is being supported financially by Rational Software. Combined with his experience as a consultant to industry, and recent award of a Royal Society Industry Fellowship (to work with IBM on modelling e-business systems), this makes him well placed to transfer results of the proposed work to industry, including the ongoing UML standardisation effort and OMG initiatives on model driven development. Kent is on the programme committee for various international conferences, including the International Conference on UML and the IEEE symposium on Visual Languages and Formal Methods. He was conference chair for «UML»'2000 [8].

Kent is supported by **Dr Peter Rodgers** and **Prof Simon Thompson**, to provide expertise in developing tools to support diagrammatic systems and in automated reasoning, respectively. **Rodgers** is a Lecturer and member of the theory group. His current research involves developing and analysing visual tools for editing, laying out and rewriting diagrams. His most recent work includes the graph drawing by graph rewriting system supported by a recent EPSRC grant (GR/M23564) [38,37], which builds on his background in diagram tools, including the production of experimental diagrammatic visualisation systems for natural language processing [11] and novel visual languages for querying graph databases [36]. Rodgers has related research in the area of novel genetic algorithm based graph layout techniques [21]. **Thompson** is Professor of Logic and Computation and leads the theory group. His work has included logical modelling of functional programming languages and machine assisted verification of functional programs [39,20]. An interest in applying formal methods has led to the application of temporal logic to the modelling of multimedia systems [3,40] and the use of semantic tableaux as decision procedures for these logics [4]. Current EPSRC funding [GR/M37851] supports his work on integration of computer algebra systems and reasoning tools [34,41].

The UKC *named researcher* is **David Akehurst**. His PhD thesis has been recently accepted, subject to minor corrections, and he is now employed on a limited contract as a research fellow by the University. An aspect of Akehurst's PhD [1] involved the definition of visual languages using an OO meta-modelling approach [2], and the partially-automated development of editors for those languages from such a definition.

The UKC team will also benefit from the expertise of **Steve Cook** who is a Visiting Professor at UKC. Cook is a Distinguished Engineer in IBM and a member of the IBM Academy of Technology, a group of 300 of IBM's top technical leaders from around the world who are working in research, hardware and software development, manufacturing, applications, and services. Cook co-developed the Syntropy OO method [6]. He has been involved in the development of the Unified Modeling Language (UML) since its creation (he was responsible for the inclusion of OCL), and represents IBM in the development of new versions of that and related standards. Cook will bring an industrial perspective to the proposed research, and will help disseminate results within IBM and in industry forums, such as the OMG.

Selected Publications

The following list contains selected publications from the project team; a further list of general references can be found at the end of the case for support.

1. **Akehurst D** (2001) Model Translation: A UML-based specification technique and active implementation approach; Ph.D Thesis, Dept of Computer Science, University of Kent.
2. **Akehurst D**, (2000) An OO visual language definition approach supporting multiple views (extended abstract), in Proc. IEEE Symposium on Visual Languages (VL2000), Seattle, IEEE Computer Society Press.
3. Bowman H, Cameron H, King P, **Thompson S** (2000) Specification and Prototyping of Structured Multimedia Documents Using Interval Temporal Logic, in Advances in Temporal Logic, H Barringer et. al., eds, Kluwer Academic Publishers.
4. Bowman H, **Thompson S** (1998) A tableau method for interval temporal logic with projection, in TABLEAUX'98, International Conference on Analytic Tableaux and Related Methods, LNAI 1397, pages 108-123. Springer-Verlag.
5. Clark T, Evans A, **Kent S**, Brodsky S, Cook S (2000) A Feasibility Study in Rearchitecting UML as a Family of Languages using a Precise OO Meta-Modeling Approach, Version 1.0. September 2000, available from www.puml.org.
6. **Cook S** and Daniels J (1994) Designing Object Systems: OO modelling with Syntropy, Prentice Hall.
7. Evans A and **Kent S** (1999) Core Meta-Modelling Semantics of UML: The pUML Approach. In Procs UML'99. IEE Press.
8. Evans A, **Kent S**, Selic B (eds.) (2000) Proceedings of «UML»'2000, LNCS 1939, Springer Verlag.
9. **Flower J** (2000) Generating Constraint Diagrams, MSc dissertation, University of Brighton.
10. **Flower J, Howse J** (2001) Generating Euler Diagrams, submitted to Graph Drawing 2001.
11. Gaizauskas R, **Rodgers P**, Humphreys (2001) Visual Tools for Natural Language Processing, To appear in The Journal of Visual Languages and Computing, 38 pages.
12. Gil J, **Howse J**, **Kent S** (1999) Constraint Diagrams: a step beyond UML, Proc. TOOLS USA 1999, IEEE Computer Society Press, 453-463.
13. Gil J, **Howse J**, **Kent S** (1999) Formalising Spider Diagrams, Proc. IEEE Symp on Visual Languages (VL99), IEEE Press, 130-137.
14. Gil J, **Howse J**, **Kent S** (2000) Advanced Visual Modelling: Beyond UML, tutorial presented at ICSE'2000, TOOLS Europe 2000, ECOOP'2000, and VL 2000.
15. Gil J, **Howse J**, **Kent S** (2001) Towards a Formalization of Constraint Diagrams, submitted to Visual languages and Formal Methods (VLFM) 2001.
16. Gil J, **Howse J**, **Kent S**, **Taylor J** (2000) Projections in Venn-Euler diagrams, Proc. IEEE Symposium on Visual Languages (VL2000), Seattle, IEEE Computer Society Press, 119-126.
17. Gil J, **Howse J**, Tulchinsky E (2000) Positive semantics of projections in Venn-Euler diagrams, Proc. Diagrams 2000, Edinburgh, LNAI 1889, Springer-Verlag, 7-25.
18. Gil J, **Howse J**, Tulchinsky E (2001) Positive semantics of projections, accepted for the Journal of Visual Languages and Computing. To appear.
19. Gil J, **Kent S** (1998) Three Dimensional Software Modelling. In Proceedings of ICSE98. IEEE Press.
20. Hill S, **Thompson S** (1995) Miranda in Isabelle. In Lawrence C. Paulson, editor, Proceedings of the first Isabelle Users Workshop, No. 397 in University Of Cambridge Computer Laboratory Technical Reports Series, pages 122-135.
21. Hobbs M, **Rodgers P** (1998) Representing space: A hybrid genetic algorithm for aesthetic graph layout, FEA'98: Frontiers in Evolutionary Algorithms, Appears in Proc. JCIS'98 volume 2, pages 415-418.
22. **Howse J**, **Molina F**, Shin S-J, **Taylor J** (2001) Type-syntax and Token-syntax in Diagrammatic Systems, submitted to Formal Ontology and Information Systems FOIS 2001.
23. **Howse J**, **Molina F**, **Taylor J** (2000) A sound and complete diagrammatic reasoning system, Proc. Artificial Intelligence and Soft Computing (ASC 2000), Banff, 402-408.
24. **Howse J**, **Molina F**, **Taylor J** (2000) On the completeness and expressiveness of spider diagram systems, Proc. Diagrams 2000, LNAI 1889, Springer-Verlag, 26-41.
25. **Howse J**, **Molina F**, **Taylor J** (2000) SD2: A sound and complete diagrammatic reasoning system, Proc. IEEE Symp on Visual Languages (VL2000), IEEE Press, 127-136.
26. **Howse J**, **Molina F**, **Taylor J**, **Kent S** (1999) Reasoning with Spider Diagrams, Proc. IEEE Symposium on Visual Languages 1999 (VL99), IEEE Press, 138-147.
27. **Howse J**, **Molina F**, **Taylor J**, **Kent S**, Gil J (2001) Spider Diagrams: A Diagrammatic Reasoning System, accepted for the Journal of Visual Languages and Computing. To appear.
28. **Kent S** (1997) Constraint Diagrams, Procs OOPSLA 97.
29. **Kent S**, Gil J (1998) Visualising Action Contracts in OO Modelling. In IEE Proceedings: Software, 2-3 in 145, 70-78.
30. **Kent S**, **Howse J** (1999) Mixing Visual and Textual Constraint Languages, Proceedings of UML99.
31. **Kent S**, **Howse J** (2001) Constraint Trees, in Clark A., Warmer J. (eds.) Advances in Object Modelling with OCL, Spinger Verlag (to appear)
32. Lauder A, **Kent S** (1998) Precise Visual Specification of Design Patterms. In Procs ECOOP98, 114-134. Springer.
33. **Molina F** (2001) Reasoning with Extended Venn-Peirce diagrammatic Systems. PhD thesis, University of Brighton.
34. Poll E, **Thompson S** (2000) Integrating Computer Algebra and Reasoning through the Type System of Aldor, in H Kirchner and C Ringeissen, editors, Frontiers of Combining Systems: Frocos 2000, LNCS 1794, 136-150. Springer.
35. **Rodgers P** (1998) A Graph Rewriting Programming Language for Graph Drawing, VL98: Proc. of the 14th IEEE Symposium on Visual Languages, IEEE Computer Society.
36. **Rodgers P**, King P (1997) A Graph Rewriting Visual Language for Database Programming, The Journal of Visual Languages and Computing, 8(6) pp. 641-674.
37. **Rodgers P**, Vidal N (1999) Pragmatic graph rewriting modifications, VL99: 1999 IEEE Symposium on Visual Languages, pp. 206-207. IEEE Computer Society.
38. **Rodgers P**, Vidal N (2000) Graph Algorithm Animation with Grrr, Agtive99: Applications of Graph Transformations with Industrial Relevance, LNCS 1779 pp. 379-394. Springer.
39. **Thompson S** (1995) A Logic for Miranda, Revisited, Formal Aspects of Computing, (7), March 1995.
40. **Thompson S** (2000) Constructive Interval Temporal Logic in Alf, in Advances in Temporal Logic, H Barringer et. al., eds, Kluwer Academic Publishers.
41. **Thompson S** (2000) Logic and dependent types in the Aldor Computer Algebra System, in Procs Calculemus 2000, M Kerber and M Kohlhase eds, A.K. Peters (to appear).

Reasoning with Diagrams: Proposed Research

1. Introduction

The problem that this research proposal addresses is how to reason with a combination of diagrammatic and textual constraint notations, in the context of modelling software intensive systems. Declarative, constraint-based languages are becoming increasingly important, as organizations struggle to define high level models capturing policy constraints and business rules. Furthermore, the preferred approach in industry to defining modelling languages is to use a meta-modelling approach. This has come to mean that definitions are expressed as object models using a combination of class diagrams and constraints, the latter to express well-formedness conditions. Examples of languages defined in this way include the Unified Modelling Language (UML) [60] and emerging e-business languages [46]. Meta-modelling will increase in importance, as domain specific modelling languages proliferate (see e.g. all the work on UML profiles at the OMG [69]).

Experience suggests that practitioners find constraints hard to read, write and analyse, and this can lead to incompleteness and inconsistency of models. Opinion is divided on why this might be. It certainly seems that different people prefer different styles of notation (mathematical, textual, visual) depending on their background. The mathematical style of notation typically used in formal methods has sometimes been blamed for their limited uptake in industry, who, it is said (see e.g. [71]), tend to prefer visual and/or programming style notations. Certainly, we agree with Parnas [62] that more attention needs to be paid to notation. Another reason could be the limited availability of *useable* tools for analysing models with constraints. Making an analysis tool useable is not just a matter of ensuring it is well tested and providing a friendly GUI. There are other factors at play. For tools to be accepted they must work in harmony with notations in widespread use, such as the UML. In particular, it must not be necessary to learn a completely different language, such as an underlying mathematical notation, to work with the analysis tool; feedback should be provided through the notations that the modeller is using.

The focus of the proposed research is a set of textual and diagrammatic notations for expressing constraints on object models. The main body of the programme will focus on notations suitable for expressing static constraints. There is a final objective to consider the feasibility of extending the framework to notations for expressing dynamic constraints. We will apply techniques drawn from the diagrammatic reasoning and tableau communities to develop sound and complete systems for reasoning with the notations in isolation and in combination. The two significant challenges here are to develop rules that work directly through the diagrammatic notations, not by translation to some underlying textual notation (see e.g. [25,22] for why these are different); and to develop techniques that support reasoning with the notations in combination. We will also develop a family of prototype tools based on the reasoning systems developed; here effort will be focussed on ensuring that reasoning is directed through the notations that the modeller is using, and that feedback is also provided through these notations. The work will be evaluated both analytically and through industry case studies and user trials.

The work is timely for two reasons. Constraint notations are becoming increasingly important in modelling software systems, and in defining the modelling languages themselves (meta-modelling); there is growing interest in model driven approaches to software development. The interest in model driven approaches is exemplified by the OMG's recent adoption of Model Driven Architecture (MDA) [68] as its strategic technical framework. This proposes to drive systems development from platform-independent UML-like visual models. To succeed, it will require modelling languages to be precise and supported by powerful analysis techniques and tools; but not at the cost of rendering those languages inaccessible and unintuitive to practitioners.

The proposed team is ideally positioned to carry out the work. The investigators have a strong track record in the main areas of expertise required: diagrammatic reasoning, visual modelling, logic and automated reasoning, tools for diagram manipulation and visualisation. The named RA's have recently completed their PhD theses in areas directly relevant to the proposal. Through Cook and Kent at UKC, the team has unrivalled access to practitioners in industry, that will both inform the research and allow the results to be disseminated quickly and effectively.

2. Scientific and Technological Background

Diagrammatic reasoning

Diagrams have always been used informally in the context of software modelling Although some notations, such as statecharts and petri nets have received more formal treatment, this tends to be focussed on the use of diagrams to define and visualise operational behaviour. Work on reasoning about diagrams expressing logical constraints has emerged from renewed interest in diagrams for expressing set-theoretic properties. In 1994, Shin [67] demonstrated that diagrammatic reasoning systems could be provided with the logical status of sentential systems. She presented formal systems of Venn-Peirce diagrams which admit purely diagrammatic reasoning and she proved these systems to be both sound and complete. We extended this work to *spider diagrams* [13,27], which are a subset of the constraint diagram notation, and an extension of the Venn-Peirce systems investigated by Shin. Spider diagrams are given model-theoretic interpretations, defining semantic functions that interpret diagrammatic elements as sets or set elements. Sound and complete diagrammatic inference rules have been developed for several systems of spider diagrams [23,24,25,26].

The proposed research will extend this work to *constraint diagrams* [28,15], which, using arrow notation, extend spider diagrams to show relations between sets and their elements. They also include symbols for universal and existential quantification over elements of sets. These extensions are required for the resulting notation to be useable for practical modelling purposes. For example, constraints written in the context of object-oriented models make frequent use of *navigation expressions* [71]. Arrow notation in constraint diagrams is used to visualise navigation expressions [28]. The proposed research will discover whether the techniques used to reason about diagrams expressing only set-theoretic properties can be adapted to more expressive notations such as constraint diagrams.

Semantic tableaux and model checking

There are at least two ways of exploring the meaning of a model: reason about the consequences, using some set of reasoning rules, such as the diagrammatic reasoning rules introduced above; and by exploring examples (instances) and counter-examples. The latter involves both checking supplied examples against a model, and generating examples. Checking is relatively simple to effect. A mechanism for generating examples is semantic tableaux. When using tableaux to check the validity of a formula, one can read off a counter-example in the case of failure [59]. A tableau can also be used to construct examples which do satisfy the model. Our programme will develop tableau systems alongside the diagrammatic reasoning rules to support analysis by example.

Another approach to example generation is model checking [44]. This technique has been successfully applied to an object modelling language [55], although its use has required some limits on what can be expressed in the language. Our programme will explore the applicability of using such techniques with the mix of notations being considered, focusing, in particular, on the practical implications of limiting expressiveness and requirement that feedback of any analysis should be provided through the notations being used.

Reasoning & visualisation tools

The programme will develop prototype tools based on the two styles of reasoning system. Tools to support the first style of reasoning fall into two broad categories: those which assist a user to build a proof [49,53] using the designated rules and those which construct proofs automatically [61,72]. We will consider both. A variety of tools founded on semantic tableaux are available as both stand-alone systems and as components of larger reasoning packages [45]. Tools to support model checking are also available and have been used to analyse object models [55].

Most work on automated reasoning tools has focused on mathematical, text-based notations. The focus in this project on diagrammatic notations brings with it complications that are not evident in a pure text-based approach. Specifically, we believe that analysis tools will be of most practical value if they can be manipulated through the notations employed by the modeller. Feedback of results must also be delivered through those notations. This means that account must be taken of issues such as diagram visualisation, including layout, and editing.

There are various diagram editing frameworks available, for example Graphlet [52]. They each offer various combinations of facilities and flexibility. Recent work in UML diagram layout such as [63,66] has particular relevance to this proposal, as does the work in displaying set based information and Venn diagrams, such as [50,56].

Generic frameworks will not by themselves provide a complete solution in supporting diagrammatic reasoning. They will have to be tailored to take account of the specifics of the languages being used, including rules on what constitutes a well formed expression. For example, (bad) experience of drawing constraint diagrams in generic tools has led to the development of a specialised editor [48] which has inbuilt knowledge of the notation. It will also be important to support an incremental approach to parsing/production [47,1], so that incremental changes to either a concrete expression or its abstract representation can be reflected in the other dynamically.

Industry modelling notations

The inspiration for constraint diagrams emerged from a desire to express, in a visual way, constraints on object-oriented models that hitherto could not be expressed *visually* using existing notations such as those found in the UML. They have been designed to work in combination with UML notations.

As well as providing a treatment of constraint diagrams as a modelling notation in their own right, our programme will use them to provide a bridge between the work on diagrammatic reasoning and notations used in industry for software modelling. Specifically, we will focus on the relationship between constraint diagrams and various UML notations currently used to express constraints on object models. The *Object Constraint Language* (*OCL*) [71,60] is a textual notation that is part of the UML standard [60]. It is intended to be a precise language mainly for the expression of invariants and pre/post conditions. Some work has been done on the formalisation of OCL, for example [64], and there are now both commercial [42] and research tools [65, 54] to support it. [42] treats OCL as a query language onto a database, [65] checks examples (object diagrams) to see if they satisfy OCL constraints, [54] type checks OCL and generates code for checking constraints in Java as part of the testing process.

There are some issues concerned with the definition of OCL that are being addressed as part of the revision of UML to version 2 [70]. This should take account of the many issues raised in attempts to formalize the language, and provide better integration with UML.

All of this work assumes the current concrete notation, which claims to be more accessible to practitioners than the mathematical symbols typically employed in formal methods. There is only hearsay evidence that the OCL syntax is more accessible. Many experienced constraint writers have complained that it is unnecessarily asymmetric and verbose. Apart from [65] and [42], little work has been done on the analysis of models involving OCL constraints. These tools only support checking of manually produced examples against such models.

OCL constraints are written in context of a UML *class diagram* [60], which provides both a vocabulary (classes and associations) and constraints on the cardinality of links that any object may have through a particular kind of association. An *object diagram* shows a particular configuration of objects at a particular point in time. A UML *collaboration* (which should not be confused with a collaboration diagram) on the other hand, attempts to specify additional constraints on possible configurations of objects that can not be specified by a class diagram. It does this by introducing the notion of roles. UML collaborations are intended to support the specification of object interactions by providing a structural description of the participants involved in that interaction. The use of collaborations to visualise aspects of OCL constraints is discussed in [43].

Constraint trees

Experience of using constraint diagrams, object diagrams and collaborations suggests that there are some constraints that are expressed much more concisely and intuitively using them. However, it has also highlighted properties that are, at best, awkward to express, without further textual annotation [30] This has led to ideas on how to use the visual notations in combination with textual languages and each other using *constraint trees* [14,31]. Constraint trees provide a modular framework in which to combine constraint notations. The nodes of a constraint tree can be logical assertions, in any notation, or logical connectives. This allows the notation to be scaleable. Constraint trees allow an ordering to be applied to the nodes; this is important in resolving some problems in constraint diagrams involving the ordering of quantifiers and issues of circularity. [31] shows how object diagrams can be used to visually express some aspects of a constraint, by using constraint trees to embed object diagrams in OCL constraints. It also explains how constraint trees provide a scaleable mechanism for organising a constraint space, by collapsing and expanding nodes.

The programme will use constraint trees as a vehicle for developing systems that support reasoning using a combination of diagrammatic and textual constraint notations.

Dynamic constraints

Although the main part of the programme focuses on notations for expressing static constraints, a final objective is to investigate the feasibility of adapting the reasoning framework to notations for expressing *dynamic* constraints. Again, the notations will involve a combination of standard UML notations (state and interaction diagrams) with more advanced proposals. For example, [29,19] show how constraint diagrams can be incorporated into 3D languages to express constraints on dynamic behaviour. [29] explains how a pair of constraint diagrams may provide the top and bottom faces of a *contract box*, which can be used to express the pre/post conditions on an operation or action. [19] shows how these boxes may be stacked up to specify algorithms and complete traces of behaviour. [19] also shows how UML state and sequence diagrams can be regarded as providing a filtered perspective on the richer 3D models.

We are also aware of related ongoing work in the recently funded EPSRC project [GR/R16891] at Edinburgh (Bradfield and Stevens). One aim of this project is to provide temporal and concurrent extensions to OCL, within the style of the current OCL syntax. We will communicate with this team to investigate the feasibility of whether/how the proposed extensions could be combined with notations for visualising dynamic constraints, and supported by reasoning systems such as those described in this proposal.

3. Programme

Aim & objectives

The aim of the research programme is to develop a framework to support reasoning with a combination of diagrammatic and textual constraint notations, suitable for use by practitioners. The specific objectives are:

(i) To develop sound and complete systems of rules for individual diagrammatic and textual constraint notations.
(ii) To develop a framework to support reasoning about constraints expressed using a combination of notations.
(iii) To prototype a family of tools to support reasoning with a combination of notations.
(iv) To establish the feasibility of extending the formal framework and tools to handle dynamic constraints.

The individual notations that will be considered are: OCL, UML class diagrams, UML Object diagrams, UML collaborations and constraint diagrams. OCL is the textual constraint language that is part of UML. OCL expressions appear in the context of a class diagram. A class diagram can also be used to impose cardinality constraints on associations. Object diagrams and collaborations can be used to notate prototypical examples, providing a limited, though diagrammatic, alternative to some aspects of OCL. Constraint diagrams visualise a significant subset of OCL and provide a direct link to ongoing work in the diagrammatic reasoning community. In addition, the programme will consider constraint trees to provide a vehicle for combining and interchanging the other notations.

Evaluation

The programme will be evaluated by the degree to which the main three objectives (i)-(iii) have been met. In addition, separate work items are included in the work plan to evaluate the usability of the notations, reasoning systems and tools. The fourth objective will be addressed depending on the success in tackling objectives (i)-(iii) and on the results of the usability evaluation.

Work plan

The research is split into a number of work items for each objective. There are two further work items to evaluate the usability of the framework that will be developed. In the descriptions below, our approach to tackling each work item is outlined. It assumed that there will be an element of dissemination (writing papers, attending conferences etc.) involved with each work item. A separate section discusses the relative timing of work items and allocation of staff resources.

(i) To develop sound and complete systems of rules for individual diagrammatic and textual constraint notations.

a) Formalise individual notations. We will continue with the approach adopted to formalise spider and constraint diagrams [13][15], but taking more care to distinguish between concrete and abstract syntax (respectively token and type syntax in [22]). Semantics will be provided by a mapping from abstract syntax to a semantics domain that is common to all notations. We will also define concrete diagrammatic and textual representations for semantics domain elements. This will allow examples to be presented concretely. We will define OCL in a similar way, noting that its concrete syntax is textual. Our formalisation of OCL will be based on recent work in this area [64].

b) Develop reasoning rules for individual notations. We will use similar techniques developed for spider diagrams to develop systems of rules for each of the diagrammatic notations. We will prove soundness, and completeness where possible. It is unlikely we will be able to develop complete systems for object diagrams and collaborations, which are notations suitable only for visualising certain aspects of constraints. We do not expect significant problems in defining a set of rules for OCL, which will be similar to the rules for FOPL.

c) Develop tableau rules for individual notations. We will adapt existing tableau systems for FOPL to OCL. Developing tableau systems for diagrammatic notations will be more challenging, as we would like the rules to work directly through the diagrams, not by translation to some underlying textual notation.

(ii) To develop a framework to support reasoning about constraints expressed using a combination of notations.

a) Formalise constraint trees. A similar approach to (i-a) will be adopted. A definition of concrete and abstract syntax will be required, which allows expressions from each of the individual notations to be plugged in.

b) Develop reasoning rules for constraint trees. A similar approach to (i-b) will be adopted. The challenge will be to define reasoning rules that allow notations nodes in a constraint tree to be expanded and collapsed, and that allow constraints expressed in one notation to be transformed into another.

c) Develop tableau rules for constraint trees. As (ii-b), though, of course, following the approach of (i-c).

(iii) To prototype a family of tools to support reasoning with a combination of notations.

a) Develop viewers and editors for individual notations, and the semantics domain. We will encode the definitions of the concrete and abstract syntax for each notation, and implement the mapping between them using a dynamic, incremental approach. Visualisation and interaction with the concrete syntax will be implemented using an appropriate diagram editing framework. We will also encode an abstract representation of the semantics domain for the notations, a concrete notation for that domain, and the mapping between the two. A viewer/editor for the semantics domain is required to support exploration by example (ii).

b) Develop tools to support reasoning with each individual notation. This will involve 3 tasks: implement the systems of rules developed in (i-b), for each notation; tie this system to the viewers and editors in (iii-a); adapt approaches to heuristics and tactics [57,58] to automate aspects of the reasoning process.

c) Develop example exploration tools for each individual notation. We will provide an encoding of the formal definition of each notation to support checking of user-provided examples against constraints written in each of the notations. Examples will be input through editors for the semantics domain developed in (iii-a). We will implement the tableau systems developed in (i-c), which can then be used as a basis for exploring how to automate aspects of example generation. We will define and implement a mapping of each notation to a form suitable for input to a model-checker, being careful to deliver feedback via the viewers (iii-a).

d) Develop viewers and editors for constraint trees. This will follow a similar approach to (iii-a), noting that the tools will need to support the ability to visualise and editing of the contents of nodes using one of the individual notations.

e) Develop tools to support reasoning with constraint trees. This will follow a similar approach to (iii-b), but using the rules defined in (ii-b). It should be possible to implement heuristics/tactics over notation interchange rules that effect automatic translation between different notations within a node.

f) Develop example exploration tools for constraint trees. This will follow a similar approach to (iii-c), but using the tableau rules defined in (ii-c).

(iv) To establish the feasibility of extending the formal framework and tools to handle dynamic constraints.

a) Define fragments of dynamic constraint notations. We will select from those notations mentioned in the background, being careful to choose some of the more risky notations to identify potential problems.

b) Define reasoning and tableau rules for these fragments.

c) Trial the implementation of editors, viewers and reasoning tools for these fragments.

(v) Evaluation

a) Analytical evaluation. We will use techniques inspired by [73] to provide an analytical evaluation of the framework produced by the project. These require the analyst to make measurements and, sometimes, judgements against a set of benchmarks designed to assess factors such as understandability and scalability. Some of the techniques are applicable to a notation on its own, and some to a notation embedded in a tooled environment. We expect that we will need to adapt and extend these benchmarks for evaluating our reasoning framework.

b) Case studies and user trials. We will use examples and case studies sourced through our industry contacts to try out the framework as it develops. In particular, we expect to use meta-modelling examples (e.g. UML 2), and examples taken from models for specifying aspects of e-business systems. The latter relates to work to be undertaken by Kent on his industry fellowship in collaboration with IBM. It is beyond the scope and resources of the proposed project to conduct a full user-trial. However, efforts will be made to obtain feedback from practitioners and students on the usability of the framework.

Resource and Time Management

The diagrammatic workplan indicates the period over which each work item is expected to run and gives some indication of which work items will be resourced by which teams. The work items have been carefully ordered as follows:

- The work items are staggered, where, discounting the evaluation work items which will be ongoing for most of the project, no site will be putting significant resource into more than three concurrent items.
- Although not shown, it is assumed that work items will tail off as new items start up.
- The ordering of work items observes the logical dependency between items. Items overlap to enable cross-fertilisation between related tasks (e.g. work on reasoning tools overlaps with the definition of reasoning rules).
- Work on reasoning rules can begin early, as appropriate definitions of some notations, which do not yet have complete reasoning systems, already exist. For

example, constraint diagrams are at least partially defined [15].

- Work on tools can begin from the start of the project, as some notations are already defined sufficiently for visualisation and editing tools to be constructed [(iii-a)]. It will also be necessary to get together resources such as a diagram editing framework.

Staff will be allocated to work items, balancing the particular skills they bring to the project, the amount of time they can spend, and the need to communicate results between the two university teams. One possible allocation is given in the table below.

(i-a)	UB team (**Howse**), Kent (UKC)
(i-b)	UB team (**Taylor**)
(i-c)	**Thompson** (UKC), Molina, Howse (UB)
(ii-a)	UB team (**Howse**), Kent (UKC)
(ii-b)	UB team (**Taylor**)
(ii-c)	**Thompson** (UKC), Molina, Howse (UB)
(iii-a)	Akehurst, **Rodgers** (UKC), Flower (UB)
(iii-b)	Akehurst, **Kent**, Rodgers (UKC), Molina (UB)
(iii-c)	Akehurst, **Kent**, Thompson (UKC)
(iii-d)	Akehurst, **Rodgers** (UKC), Flower (UB)
(iii-e)	Akehurst, **Kent**, Rodgers (UKC), Molina (UB)
(iii-f)	Akehurst, **Kent**, Thompson (UKC)
(iv-a)	UB team (**Howse**), Kent (UKC)
(iv-b)	UB team (**Howse**), Thompson (UKC)
(iv-c)	UKC team (**Kent**), Flower, Molina (UB)
(v-a)	**Flower**, Molina (UB)
(v-b)	Akehurst, **Kent**, Cook (UKC)

Here, Thompson's and Rodger's efforts are focussed in areas where they can contribute most effectively, tableau/automated reasoning and diagram viewers/editors, respectively. As the architect of some of the notations, Kent is also involved in their formalization. As someone with expertise in visualising constraint diagrams, Flower will contribute to the viewer/editor work. Some work items involve members from each team, thereby facilitating the transfer of ideas. Individuals in the UB team are more interchangeable than those in the UKC team; therefore the UB team has been allocated to some work items without further distinction. The person leading each work item is shown in bold. In most cases, this person comes from the team providing most of the resource. There is one exception: Thompson leads the work on developing tableau rules, as he has most expertise in this area; however, UB, in particular Molina the RA, will put in most effort.

The two principal investigators, who have worked together successfully before, will lead the project at the two sites, and overall project management will be the responsibility of Dr Howse, who has undertaken this role on two previous EPSRC grants *Formal Underpinnings of Object Technology* (GR/K67304, 1995-1999, also a two-site project) and *Developing and using formal models of inheritance* (GR/H16629, 1992-1995). Both projects received the grade *good* for management and use of resources.

Although Cook is not formally part of the team (he is a visiting professor at UKC), he will contribute to the industrial evaluation of the work through his position in IBM. We aim to continue our collaboration with Dr Yossi Gil from the Technion, Israel, who has worked with us on constraint diagrams and on the development of 3D notations.

4. Relevance to Beneficiaries

There are four main communities who will benefit from the research:

- *The Modelling community (including the UML and Meta-Modelling communities).* The project will provide a framework for OO modelling using a mixture of visual and textual constraint notations. This will be supported by prototype reasoning and visualisation tools. The results will push forward current thinking in this community on what it is possible to express visually, and in the kinds of (meta-)modelling tools it is possible to construct. Our experience with this community is that theory is more likely to be understood and adopted if supported by tools.
- *Diagrammatic Reasoning community*. The project will demonstrate whether and how the mathematical techniques employed by this community scale up to modelling notations required in practice.
- *Automated Analysis and Reasoning communities.* The project will show whether and how reasoning tools can be driven through diagrammatic notations, in isolation or in combination with each other and with textual notations.
- *Users and developers of software intensive systems.* By using real examples provided by our industrial contacts, we will be able to show the potential benefits and limitations of our techniques and tools in dealing with models of industrial-scale software intensive systems. In the medium term, successful exploitation of the results of the project will lead to more accurate and complete models of software systems, and, through meta-modelling, of standards such as UML. This should lead to systems that are more reliable and fit for purpose, which can be developed more rapidly through a model driven approach.

5. Dissemination and Exploitation

The usual channels will be used for academic publication including the conferences, Diagrams, VLFM, CADE, TABLEAUX, TPHOLs, LICS, ETAPS, OOPSLA, ECOOP, UML, TOOLS, ICSE, BMC, BCTCS; and journals, Journal of Visual Languages and Computing, FACS, LMS Journal of Mathematics & Computation, Journal of Logic and Computation, Theoretical Computer Science, IEEE TSE.

Our intention is to make the prototype tools freely available under an appropriate license. This is standard academic practice, and is being adopted in some industry quarters (e.g. IBM's alphaworks site). We will also provide demonstrations at conferences and to our industrial contacts.

The work will be disseminated to the open standards community in modelling languages, through Cook and Kent's close involvement with standardization and revision of UML. Kent's industry fellowship and Cook's position in IBM will provide additional routes for the dissemination of the work to industry.

The ideas behind the tools should be exploitable, but only as part of a larger venture in developing better commercial tools to support modelling. Such a venture could be in partnership with one of our industrial contacts such as Rational or IBM. Exploitation need not be hindered by making the research prototypes freely available; we will seek advice about any patent issues that might arise before taking this step.

6. Justification of Resources

Research Staff

One RA at each site. Molina's (UB) background in developing a sound and complete reasoning system for spider diagrams is ideal for this project. Akehurst's (UKC) background in tool building, modelling and meta-modelling is also well-suited.

Support

Support for the project will be provided by 10% of an administrator at UKC, 20% of an administrator at UB (an additional 10% as this is where overall management of the project will be), and 10% of a technician at each site. A charge for computing infrastructure for each RA has also been included.

Travel

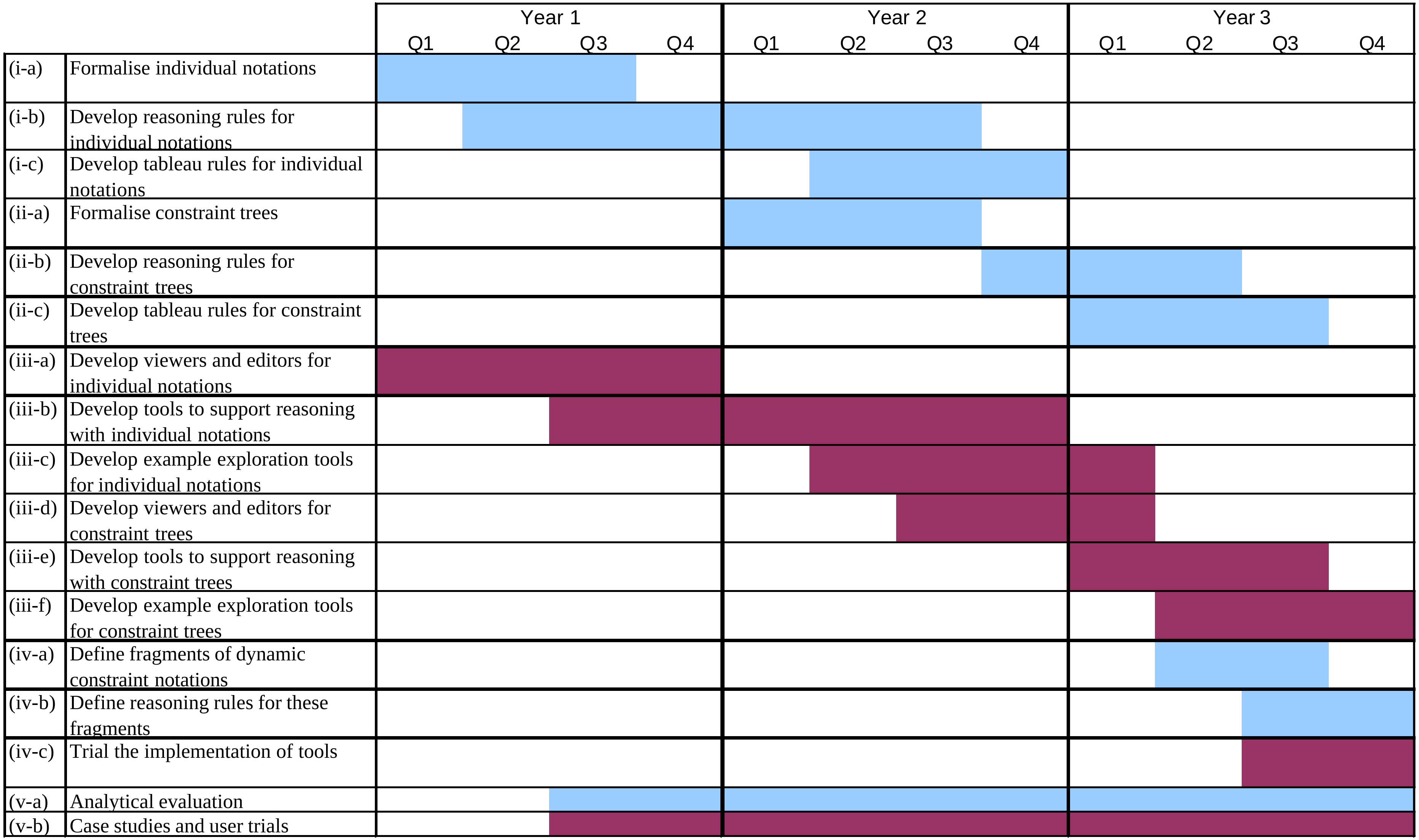
We expect the project to be successful, and therefore to generate a significant number of papers. The travel budget reflects this. The cost of conference trips has been averaged out: worldwide trips usually cost more than £1500, and European trips a little less. A modest budget has been requested to support travel between the two sites, and with other sites (e.g. Edinburgh, York, KCL) in the UK. We have requested funds for two trips (1 per site) between the UK and Israel to support collaboration with Dr Gil. We have costed these trips at the same rate as conferences.

Equipment & Consumables

- A laptop, with accompanying docking station and peripherals, for the RA at each site. Toshiba is standard issue at both sites. We consider laptops to be standard equipment for postdoc researchers, as their productivity benefits amply justify their relatively low cost. A docking station avoids the need to also purchase a separate desktop machine.
- A contribution to the cost of laptops for the investigators, at rate of 1/3rd laptop (with accompanying peripherals) per investigator. It is now standard practice to partially fund equipment for investigators from research grants.
- Software and computing supplies at rate of £1500 per machine, to include upgrades over the course of the project, and £500 for books at each site.
- Computing infrastructure charge of £1500 per year per research assistant.

7. References

42. Boldsoft (2001) The Model Run tool, available from www.boldsoft.com.
43. Bottoni P, Koch M, Parisi-Presicce F, Taentzer G (2001) A Visualization of OCL using Collaborations, submitted to UML'2001.
44. Clarke E, Gumberg O, Peled D (1999) Model Checking, MIT Press.
45. D'Agostino M (Editor) (1999) Handbook of Tableau Methods, Kluwer Academic Publishers.
46. Edifecs (2001) E-business Collaboration Modeling Metamodel, available from http://www.edifecs.com/professional_services_docs.jsp
47. Wagner T, Graham S (1998) Efficient and flexible incremental parsing. In ACM Transactions on Programming Languages and Systems, Volume 20, Issue 5.
48. Gil J, Sorkin Y (2001) The Constraint Diagrams Editor, http://www.cs.technion.ac.il/Labs/ssdl/research/cdeditor/
49. Gordon M, Melham T (1993) Introduction to Hol: A Theorem Proving Environment for Higher Order Logic, Cambridge University Press.
50. Graham M, Kennedy J, Hand C (2000) A Comparison of Set-Based and Graph-Based Visualisations of Overlapping Classification Hierarchies. Proc. Advanced Visual Interfaces (AVI) 2000. ACM Press.
51. Hammer, E.M. (1995) Logic and Visual Information, CSLI Publications.
52. Himslot M (1997) The Graphlet System, Proc. Graph Drawing (GD) 96, LNCS 1190, Springer-Verlag.
53. HOL on the web. http://www.cl.cam.ac.uk/Research/HVG/HOL/
54. Hussman H, Demuth B, Finger F (2000) Modular architecture for a toolset supporting OCL. In [8].
55. Jackson D, Schechter I, Shlyakhter I (2000) Alcoa: the Alloy Constraint Analyzer. In Proc. International Conference on Software Engineering, Limerick, Ireland, June 2000.
56. Johnson D and Pollack H (1987) Hypergraph Planarity and the Complexity of Drawing Venn Diagrams, Journal of Graph Theory, 11.
57. Jones C, Jones K, Lindsay P, Moore R (1991) Mural: A Formal Development Support System, Springer Verlag.
58. MacKenzie D. (1995) Automation of Proof: A Historical and Sociological Exploration, Annals of the History of Computing, Fall.
59. Negri S, von Plato J (1998) From Kripke Models to Algebraic Counter-valuations, in TABLEAUX'98, Herrie de Swart (ed.), Lecture Notes in Computer Science 1397, Springer Verlag.
60. Object Management Group (1999) OMG Unified Modeling Language Specification, Version 1.3., available from www.omg.org, June.
61. OTTER on the web. http://www-unix.mcs.anl.gov/AR/otter/
62. Parnas D (1996) Mathematical methods: What we need and don't need. In *An invitation to Formal Methods*, pages 16-30, IEEE Computer, Vol 29 No 4, April.
63. Purchase H, Allder J, Carrington D (2000) User Preference of Graph Layout Aesthetics: A UML Study, Proc. Graph Drawing (GD), 2000 LNCS 1984, Springer-Verlag.
64. Richters M, Gogolla M (1998) On formalizing the UML object constraint language OCL. In Proc. 17th Int. Conf. on Conceptual Modeling (ER'98), LNCS 1507, Springer-Verlag.
65. Richters M, Gogolla M (2000) Validating UML models and OCL constraints. In [8].
66. Seemann J (1997) Extending the Sugiyama Algorithm for Drawing UML Class Diagrams: Towards Automatic Layout of Object-Oriented Software Diagrams Proc. Graph Drawing (GD) '97, LNCS 1353, Springer-Verlag.
67. Shin, S-J (1994) The Logical Status of Diagrams. CUP.
68. The OMG (2001) Executive Overview: Model Driven Architecture. Available from http://www.omg.org/mda/
69. The OMG (2001) Work in progress, http://www.omg.org/schedule/
70. The OMG (2001) UML 2.0 OCL Request for Proposals, available from http://www.omg.org/uml
71. Warmer J, Kleppe A. (1999) The Object Constraint Language: Precise Modeling with UML. Addison-Wesley.
72. Wos L with Pieper G (2000) A Fascinating Country in the World of Computing: Your Guide to Automated Reasoning, World Scientific.
73. Yang S, Burnett M, Dekoven E, Zloof M (1997) Representation Design Benchmarks: A Design-Time Aid for VPL Navigable Static Representations. In Journal of Visual Languages and Computing 8, 563-599, Academic Press.



		Year 1				Year 2				Year 3			
		Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
(i-a)	Formalise individual notations	🟦	🟦	🟦									
(i-b)	Develop reasoning rules for individual notations		🟦	🟦	🟦	🟦	🟦	🟦					
(i-c)	Develop tableau rules for individual notations						🟦	🟦	🟦				
(ii-a)	Formalise constraint trees					🟦	🟦	🟦					
(ii-b)	Develop reasoning rules for constraint trees								🟦	🟦	🟦		
(ii-c)	Develop tableau rules for constraint trees									🟦	🟦	🟦	
(iii-a)	Develop viewers and editors for individual notations	🟥	🟥	🟥	🟥								
(iii-b)	Develop tools to support reasoning with individual notations			🟥	🟥	🟥	🟥	🟥	🟥				
(iii-c)	Develop example exploration tools for individual notations						🟥	🟥	🟥	🟥			
(iii-d)	Develop viewers and editors for constraint trees							🟥	🟥	🟥			
(iii-e)	Develop tools to support reasoning with constraint trees									🟥	🟥	🟥	
(iii-f)	Develop example exploration tools for constraint trees										🟥	🟥	🟥
(iv-a)	Define fragments of dynamic constraint notations										🟦	🟦	
(iv-b)	Define reasoning rules for these fragments											🟦	🟦
(iv-c)	Trial the implementation of tools											🟥	🟥
(v-a)	Analytical evaluation			🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
(v-b)	Case studies and user trials			🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥

mostly resourced from UB 🟦 mostly resourced from UKC 🟥