

CASE Tools and Constraints

R J Offen

CSIRO-Macquarie University Joint Research Centre for Advanced Systems Engineering (JRCASE)
Macquarie University, North Ryde, NSW 2109, Australia
roffen@ics.mq.edu.au

Abstract

This workshop report briefly describes the CADPRO project and its goals, and then reports on the early phases of research that were necessary before the empirical studies central to the project could be satisfactorily designed, planned and executed. The report discusses, in outline only, the CADPRO research relating to the identification and characterisation of constraints in CASE design tools and the related development of the JRCASE metaCASE toolset, *CASEMaker*, as well as some of the lessons learnt from the two strongly formative CADPRO pilot studies, which were important precursors to the full-scale experiments.

Keywords: CASE tools, Design, Productivity, Quality, Constraints, Pilot studies

1. Introduction

CADPRO (Constraints And the Decision **PRO**cess) is an international collaborative research project involving The University of New South Wales (UNSW), Macquarie University (MU), and the Nara Advanced Institute of Science & Technology (NAIST). The project, which started in 1996, has several goals: to build close, productive, collaborative links between research centres in Australia and Japan; to demonstrate novel techniques for building flexible software design tools; and to refine the collaborators' already considerable capabilities and experience in empirical software engineering research.

Prior research¹, based upon self-report techniques, had already established that user perceptions of high constraint are associated significantly with low satisfaction and with resistant behaviour while using a computerised design tool. This research also indicated that favourable attitudes toward control are associated significantly with

behaviour which conforms to the apparent intent of constraints. Based on this prior work the initial CADPRO research design was predicated on the 'constraints model' described in Figure 1, below.

¹ Day, D. (1994). Behavioral and perceptual responses to the constraints of computer-mediated design. In Brouwer-Janse, M. & Harrington, T. (Eds.), *Human-Machine Communication for Educational Systems Design*. Series F: Computer and Systems Sciences Vol. 129, NATO ASI Series. Berlin: Springer-Verlag.

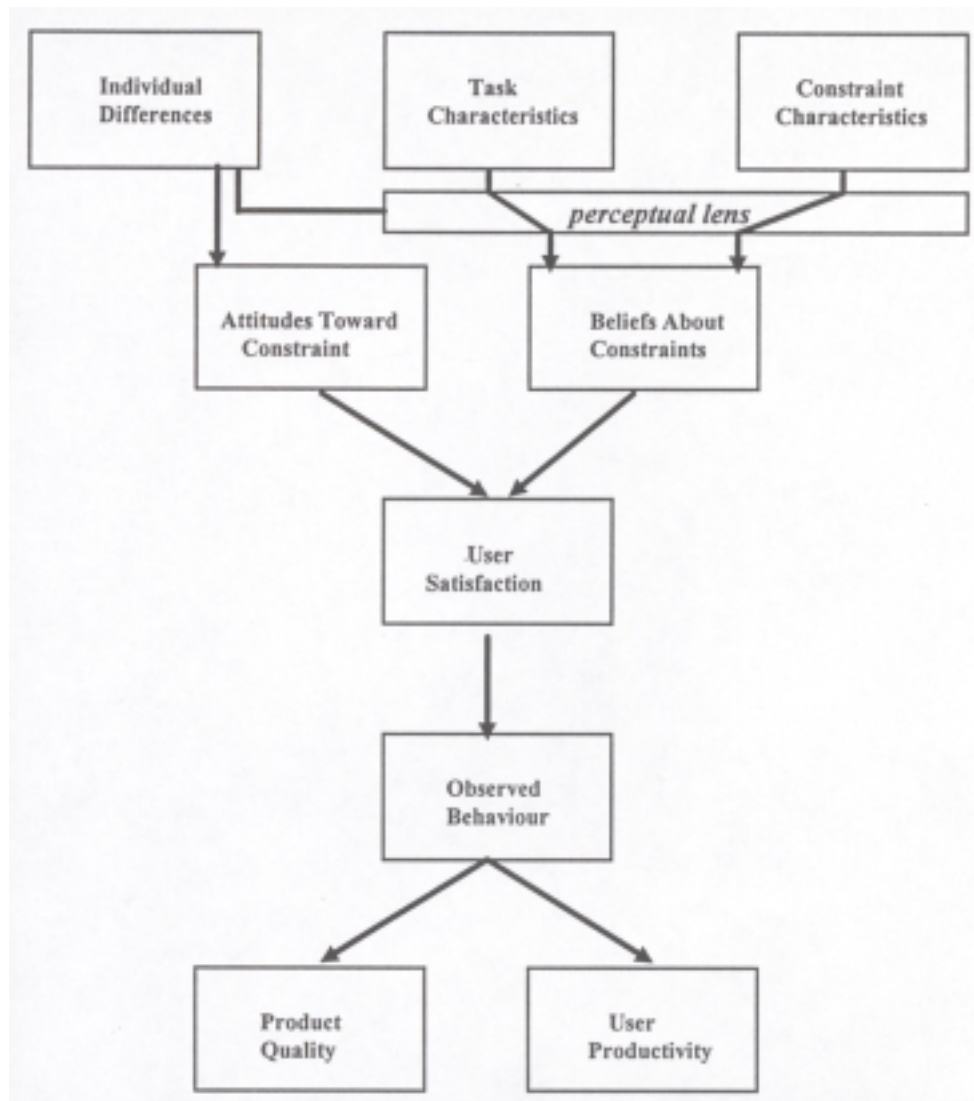


Figure 1: The CADPRO research model

The primary CADPRO hypothesis is that constraints in CASE tools significantly affect the ways in which software designers work, which in turn affects both the *quality* of the design products created and user *productivity*. In order to test this hypothesis, a more thorough understanding of the nature of constraints was required. This led to CADPRO researchers developing, early in the project, a *constraint taxonomy* that could then be used to evaluate systematically user responses to CASE design tool constraints². Suitable measures for design quality and productivity were also exhaustively investigated, in that many pre-existing measures for these attributes proved to be highly problematic when applied to the emerging experimental design. The ultimate goal of the CADPRO project is to facilitate the improved implementation of constraints in CASE tools. We believe that one reason why CASE

² Scott, L., Hovarth, L. and Day, D. (1998) Characterising Constraints on Behaviour in CASE Tools, JRCASE Research Report 98/8 and CAESAR Technical Report 98/5.

tools are not used as extensively as had been expected³ is that, in general, their implementation of constraints is arbitrary and frequently counter-productive. The tools are not flexibly adaptable to users' decision-making styles or experience, which manifest themselves in the form of 'rules' applied during software design. If practitioners were able to use more constraint-adaptive tools, product quality and user productivity might increase, effectively reducing labour costs and minimising system failures, while at the same time increasing employee morale and job satisfaction.

2. Constraints in CASE Tools

Within software and systems design, constraints play an important role in guiding the development of products. Often incorporated as part of a *development methodology*, constraints are most often manifested as rules and guidelines. Ideally, they are intended to limit the design space and therefore to guide the designer toward a 'good' solution. In this way, a methodology may provide valuable support for designers. However, due both to personal preferences and to the differing natures of projects and companies, the constraints specified by methodology are rarely followed to the letter⁴ and often prove counter-productive. Instead, designers modify methodologies, using a combination of techniques to suit their own preferences, abilities and circumstances.

The role of constraints in design methodologies is important. Constraints become paramount when a CASE tool is used to support systems development. CASE tools have the ability to *enforce* methodological constraints, making it difficult for designers to ignore or circumvent them - as they might if they were not using a CASE tool. For example, the tool may enforce a naming convention which includes ensuring that all design artefacts are uniquely named. This can be a great benefit, ensuring that designs conform to the development methodology, and to notation standards. However, it also can be a curse, contributing to a general sense of frustration with the tool and relegating it to use simply for *post hoc* design capture rather than for creative design development.

Within the context of CASE tools, a constraint is a rule that defines the range of options available in performance of a task. In the very simplest sense of the word, most tools *constrain* the user to the application of a particular modelling method, by simply not supporting alternative modelling mechanisms.

Constraints can, however, be more sophisticated. For example, constraints on the structure of models might include a rule that **every process in a data-flow model must have at least one input data flow and at least one output data flow**. Or, constraints on the sequence of design activities might specify that **a parent process must be specified before its child processes**. These kinds of constraints, referred to here as *methodological constraints* since they are specified by design methodologies, are implemented in CASE tools to guide the user toward a 'good' or 'correct' design. By instantiating particular methodologies, the builders of CASE tools attempt to control

³ Iivari, J. Why are CASE tools not used? *Communications of the ACM* 39, 10 (1996), 94-103.

⁴ ter Hofstede, A. H. M. and Verhoef, T. F. Meta-CASE: Is the game worth the candle? *Information Systems Journal* 6 (1996), 41-68.

⁵ Webster, D. E. Mapping the design information representation terrain, *IEEE Computer* 21, 12 (1988), 8-23.

the behaviour of tool users - by communicating (through the tools) the necessary and appropriate processes in software development.

Examples of constraints are listed in Table 1, drawn from a survey of seven CASE tools (Paradigm Plus, Rational Rose, MetaEdit, SilverRun, Teamwork, Toolkit for Conceptual Modelling, and WinCASE). Not conforming to such constraints may risk an incomplete or inconsistent design, or at worst one which is unusable in later stages of systems development. However, the unnecessary or inappropriate enforcement of constraints in the earlier, creative stages of design can limit the effective and productive application of developer expertise.

Table 1: Examples of Constraints

Process Modelling

- A DFD either must be a context diagram or have a parent process on a higher-level DFD
- A parent process must be specified before its child processes
- External entities must be connected only to a process
- Each data store on a set of DFDs must be uniquely named

Data Modelling

- Entity must be specified before its relationship
- Entity and relationship must be specified before their attributes
- Cardinality must be shown at each end of a relationship
- Associative entities must have one or more attributes

Event Modelling

- All states shown must be attainable
- Final state in lower-level diagram must correspond to exit in associated higher-level state
- Only one start state may be placed in each diagram
- Every diagram must have a start state and a final state

Class Modelling

- Unidirectional associations cannot be drawn between a class and a package or a node
 - An association cannot be drawn between a package and an actor
 - An aggregation cannot be drawn between a parameterised class utility and a node
 - A generalisation cannot be drawn between a parameterised class utility and a use case or a node
-

The treatment of constraints in the context of tool-assisted problem solving is a relatively new area of enquiry. Theory relevant to this area has been developed in decision support⁶, cognitive psychology⁷, software engineering^{8 9}, and human-computer interaction¹⁰.

In Reference 2, Scott, Hovarth and Day present a comprehensive taxonomy of constraints, as developed by CADPRO researchers, after a review of the studies noted

⁶ Silver, M. *Systems That Support Decision Makers*. Wiley, Chichester, UK, 1991.

⁷ Darses, F. The constraint satisfaction approach to design: a psychological investigation. *Acta Psychologica* 78 (1991), 307-325.

⁸ Jankowski, D. Computer-aided systems engineering methodology support and its effects on the output of structured analysis. *Empirical Software Engineering* 2 (1997), 11-38.

⁹ Vessey, I., Jarvenpaa, S. and Tractinsky, N. Evaluation of vendor products: CASE tools as methodology companions. *Communications of the ACM* 35,4 (1992), 90-105.

¹⁰ Day, D. User responses to constraints in computerised design tools: an extended abstract. *Software Engineering Notes* 21,5 (1996), 47-50.

above, a comprehensive survey of CASE tools, and reference to a number of systems development guidelines. Under the proposed scheme, each potential constraint in a CASE tool may be described using a predefined list of values for each taxonomy characteristic. A characteristic is a classification that may be used to group or separate constraints, based on whether they share some significant common values. In this way, a profile of each constraint may be created. This profile is comprised of the relative values for all characteristics, for that constraint.

In the CADPRO project, this taxonomy was subsequently used to guide decision making regarding the types and levels of constraints to apply as developers create design products during laboratory observations. As part of later data analysis, the degree of constraint flexibility built into the version of the tool used during observation could then be associated systematically with user productivity and with the quality of design products generated.

Future work that is envisaged includes the use of the CADPRO constraint taxonomy in the construction of a variable-constraint front-end for CASE tools. Such a product would allow constraints in CASE tools to be varied according to guidelines determined by the individual, according to the project context, or by employer policy. It may also be possible to assess the expertise of users dynamically, as tools are used. That would make possible adaptive interfaces for automated design tools - interfaces that adjust 'restrictiveness' based on design strategy/tactics, anticipated product quality and user productivity.

3. A Basis for Empirical Studies

In empirical software engineering, pilot studies typically result in relatively minor adjustments of a planned experimental design. In the CADPRO project, two pilot studies radically changed the way in which data was eventually gathered and interpreted i.e. the pilot studies were strongly rather than weakly formative. The original plan in the CADPRO project was to have professional software developers undertake analysis/design tasks in 50 minute sessions in a useability laboratory using a CASE tool generated by *CASEMaker*, a metaCASE tool¹¹. *CASEMaker* is sufficiently flexible to allow specific configuration of the methodological constraint environment. Conventional productivity and quality metrics were to be applied to the design artefacts created under different configurations of the constraint environment, thus allowing a test of the research model. On this assumption, pilot studies were planned and undertaken so as to ensure the substantive experimentation ran smoothly. The main aim of the first pilot study was to have subjects undertake analysis/design tasks, under conditions similar to those intended for the CADPRO useability laboratory experimentation, in order to be able to evaluate proposed productivity and quality metrics. Therefore, subjects worked alone, problem descriptions were no more than half a page, there was a time limit of 50 minutes, and subjects did not use pen and paper. Various decisions had to be made concerning choice of subjects and tasks. The tools to be used had also to be decided upon as *CASEMaker* was still under

¹¹ L. Scott. "Towards Flexible Conceptual Modelling in CASE Tools Using N-Graphs" (Phd Thesis). JRCASE Technical Report no. R.R. 98/16, Macquarie University Sydney, NSW 2109, Australia. 1998.

development. In this pilot study, no attempt was made to reproduce the facilities offered by an actual useability laboratory eg no video records were made.

Two problem descriptions were chosen: a Mail Order System for data-flow diagram (DFD) modelling and an Automatic Map Labelling System for class modelling. Three CADPRO researchers attempted these problems and produced corresponding DFD models and class models. This work was all undertaken on the same day at the same location (JRCASE). The Mail Order System problem was tackled first followed by a break to make notes before tackling the Automatic Map Labelling System problem.

As *CASEMaker* was still under development, other tools needed to be considered for the pilot studies. CASE tools that were currently available at JRCASE were chosen for convenience and because they were familiar to two of the investigators. Furthermore, it was decided that, for each problem, different tools were to be used to help provide an understanding of the impact of other tool issues (e.g. human-computer interface issues). The Toolkit for Conceptual Modelling¹² (TCM Version 1.1.0, Unix, May 1996) was used for DFD modelling and class modelling. A demonstration version of the MetaEdit¹³ CASE tool (PC, 1993) was used for DFD modelling. Paradigm Plus¹⁴ (Unix, Version 3.5, 1996) was used for class modelling. At the conclusion of each 50-minute work period, the investigators met to relate and make notes of their experiences.

A wide range of productivity and quality metrics were reviewed for their applicability to the artefacts produced in this first pilot study and where possible, metric values were calculated and reasons sought for any variations. The evaluation of the suitability of the proposed metrics was partly judgmental. Comprehensive metric validation requires not only access to intrinsic quality factors (such as counts of errors that result in operational failures) but also that the metrics and quality factors be shown to be statistically associated. The first pilot was successful in establishing that a traditional metrics approach to productivity and quality measurement of early life-cycle artefacts would be inappropriate for the proposed useability laboratory experiments.

Investigators making different assumptions, largely as a result of uncertain problem boundaries brought about by using short problem descriptions, caused variation in the values obtained for metrics. To a lesser extent, the different experience levels of the investigators with a particular modelling technique or tool also caused variation. The lack of explicit support of levelling in TCM was another explanation for unsatisfactory DFD models. User-interface concerns also impinged more on the work of the investigators much more than one might have expected. This was the first hint that *interface constraints* could play as much a part as methodological constraints in shaping end-user behaviour of CASE tools, and so potentially confounding the influence of these constraints on the response variables, productivity and quality.

Various recommendations were made for the next pilot, including: To reduce unwanted variability, problem descriptions should be more detailed, more time should

¹² <http://www.cs.vu.nl/~tcm/tcm.html>

¹³ <http://www.jsp.fi/metacase>

¹⁴ http://www.platinum.com/products/appdev/pplus_ps.htm

be allowed; completion times should be noted; user-interfaces should be of a high quality; and subjects should preferably be experienced with the modelling technique and tool. It was proposed that quality be assessed by reference to a sample solution (with checklists) or by a panel of experts, and that product delivery measures should be profiled against time – ie the so-called CADPRO *delivery profile* (see Figure 2 below). This last innovation, regarding 'delivery profiles', turned out to have a very significant influence on all the subsequent data collection and analysis protocols - shifting the research methodology substantially from what had been proposed at the outset.

In the second pilot study, subjects undertook analysis/design tasks modified to take account of the concerns that arose in the first pilot study. One of the main objectives was to record and interpret the delivery profiles. In contrast to the first pilot study, subjects undertook the tasks in an actual useability laboratory, part of Professor Torii's software engineering laboratory at the Nara Institute of Science and Technology. Video records were made of the computer display that included audio records of the subjects' utterances. Subjects worked alone, problem descriptions were no more than a page, there was a time limit of 2 hours, and subjects did not use pen and paper. As the *CASEMaker* tool was now sufficiently developed, subjects for the task requiring data-flow diagram (DFD) modelling used a DFD tool generated by *CASEMaker*. As recommended from the first pilot study, sample solutions were used as reference points in evaluating the solutions produced, subjects' completion times were noted, and profiles of product delivery against time provided by use of an appropriately instrumented *CASEMaker* were analysed.

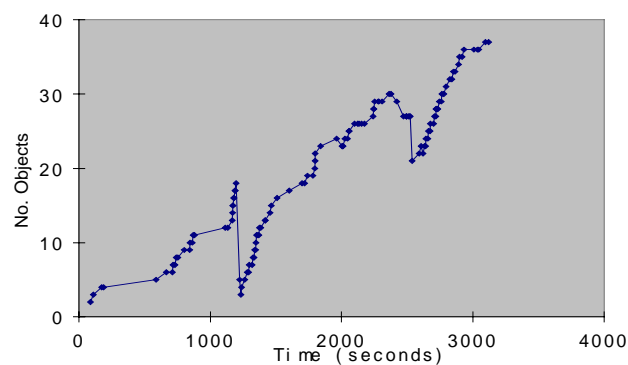


Figure 2: A typical subject 'delivery profile' that shows how the number of model nodes and edges (y-axis) evolved with time (x-axis)

As with the first pilot study, subjects were required to perform DFD modelling for a Mail Order System and class modelling for an Automatic Map Labelling System. The problem descriptions were modified to take account of concerns that arose as a result of the first pilot study. For each task, different tools were used. The *Toolkit for Conceptual Modelling* (TCM Version 1.6.6)¹⁵ was used for class modelling while *CASEMaker* was used for DFD modelling. *CASEMaker* recorded time-stamped delivery profiles for the DFD models produced by the three subjects. Creations and deletions of nodes (entities and processes) and edges (data-flows) were recorded.

¹⁵ <http://www.is.cs.utwente.nl:8080/~tcm/index.html>

Renaming actions were also recorded. The delivery profiles were visually examined for the presence of dips which may represent productivity loss.

As a consequence of the two pilot studies many problems with the subjects' problem description and instructions/briefing/training were 'ironed out'; the notion of a 'delivery profile' became the new basis for productivity assessment; and design quality was subsequently determined by an 'expert' qualitative assessment.

4. MetaCASE Technology

CASE design tools can and do play an important role in software and systems development, by providing support for designers creative, analytic and clerical tasks. However, most CASE tools fail to take into account the fact that individuals, companies and different projects have different needs for the kind of support they require from their CASE tools. To address this problem, metaCASE tools have been proposed and developed by both researchers and commercial vendors. metaCASE tools provide means by which CASE tools can be customised to suit the requirements of different users.

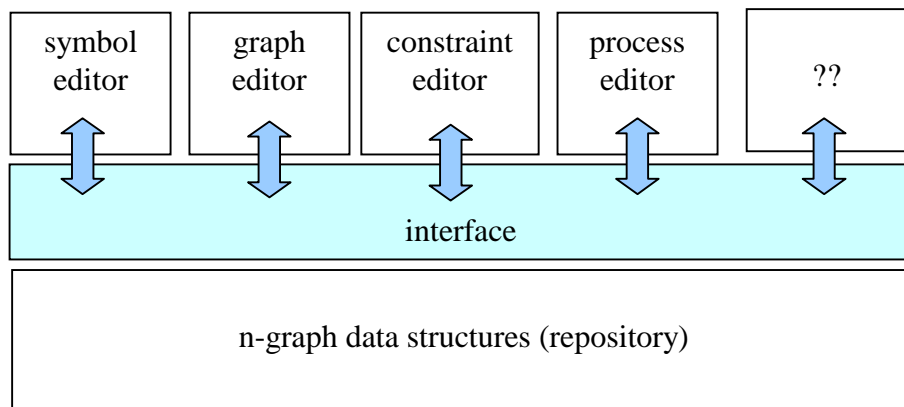


Figure 3: The architecture of CASEMaker.

The JRCASE *CASEMaker* tool is unique amongst metaCASE tools in two respects. Firstly, it uses a new meta-data model (*n-graphs*¹⁶) and an architecture designed to enhance the ability to customise CASE design tools and the ease with which this customisation can be done. Secondly, it concentrates on providing support for vendor/developer/user configurable constraints. Other interesting features of the tool include the ability to reconfigure a specifically generated design tool without the need to re-compile it, and the use of the Java *Reflect* package to allow dynamic binding of Java classes to define notation and constraint detection which enhance the flexibility of the metaCASE tool. A major research objective of the JRCASE metaCASE project was to develop and instrument a variable constraint CASE tool specifically for use in the CADPRO empirical trials.

¹⁶ A *n-graph* is a directed graph where any node can itself be an *n-graph*. Thus *n-graphs* are *nested*, directed graphs. *N-graph* morphisms are used to model relationships between *n-graphs*, eg to model an aggregation or an instantiation relationship.

Figure 3 schematically illustrates the architecture of *CASEMaker*. The bottom ‘layer’ represents the n-graph¹⁷ data structures, the layer above this is the main utility class and the top-most ‘blocks’ represent the various meta-tools associated with *CASEMaker*. These meta-tools also embody the graphical-user-interface functionality of *CASEMaker*.

The *n-graph data structures* utilise the n-graph meta-model¹⁷ as the underlying repository for the tool. The n-graph meta-model is used in *CASEMaker* as a conceptual modelling kernel which stores all of the information about the structure of the repository as well as data instances.

The *interface* functions as a link between the basic n-graph data structures and the user interface. It functions like a DBMS in that it handles operations on the repository, such as creating, deleting and renaming n-graphs (and edges). The interface layer also contains the instrumentation for recording user actions and constraint violations.

The *graph editor* functions as the main window of the *CASEMaker* tool. It provides functions for creating and editing n-graphs, as well as links to the other tools.

The *symbol editor* provides the functions for customising the notation supported by the tool. It allows symbols to be created by either importing a gif image of the symbol or by attaching a Java class for drawing the symbol.

The *constraint editor* provides the functions for specifying the nature of constraints, notification of the violation of a constraint and the reaction to a constraint violation.

The *process editor* has not been implemented in the current versions of *CASEMaker*.

The “??” section is designed to allow for the future extension of *CASEMaker*.

Significant instrumentation extensions were made to *CASEMaker* after the CADPRO pilot studies, so as to automate much of data-analysis that needed to be undertaken on what were very large and complex data sets.

5. Diagramming Notations

In the early stages of the project the predominant emphasis of the research, the pilot studies, and the associated customisation of *CASEMaker*, were very much focussed on data-flow diagrams, though some exploratory O-O design pilot trials were undertaken. It was only in the later stages of the project that the empirical emphasis started to shift towards O-O notations, facilitated by the project’s access to the source code of the *Simply Objects* CASE tool¹⁸, which turned out to be easier to make use of than to produce an O-O design tool from *CASEMaker*, the developer of which having

¹⁷ L. Scott. "Towards Flexible Conceptual Modelling in CASE Tools Using N-Graphs" (PhD Thesis). JRCASE Technical Report no. R.R. 98/16, Macquarie University Sydney, NSW 2109, Australia. 1998.

¹⁸ from Adaptive Arts Pty Ltd.

taken up a post-doctoral position in Germany by this time. The O-O aspects of the CADPRO experiments will be discussed in subsequent workshop reports.

6. Conclusions

The early stages of the CADPRO project were crucial precursors to the subsequent design, planning and execution of the main experimental studies. It is fair to say that without this early work the eventual outcomes would have been highly problematic, to say the very least.

Acknowledgments: This ‘end-of-project’ workshop paper incorporates important contributions to the CADPRO project from researchers Louise Scott, Andy Brooks, Shingo Takada, Levente Hovarth and others. Research funding was provided by the ARC and the CSIRO, and travel support by DETYA (TIL) and Macquarie University (MURG). All these contributions, individual and from funding agencies, are gratefully acknowledged, as is all the support and good advice provided, over the duration of the project, by the CADPRO external advisory board.