A Formal Model of Mixed-Initiative Interaction in Design Exploration

Sambit Datta[1] and Michael Hobbs[2]

[1] School of Built Environment,
Curtin University
sambit.datta@curtin.edu.au
[2]School of Information Technology,
Deakin University
mick@deakin.edu.au

Abstract. Computer-based environments for supporting design are complex software artifacts. These tools need to use sound computational formalisms as well as address issues of human usability. The development of interactive and usable generative systems is a significant research area in design computation. Though classical search techniques play a central role in the generative kernels of these "closed-world" systems, the open-ended exploration of design spaces is the desirable goal. In this paper, we present a formal model of exploration that combines search with user driven exploration. We describe the role of interaction and agency in an experimental mixed-initiative design support system.

Keywords: generative design, mixed-initiative, design exploration.

1 Background

Generative design comprises an iterative process of specifying problems, finding plausible and alternative solutions, judging the validity of solutions relative to problems and reformulating problems and solutions. Mixed-initiative is an effective paradigm for addressing the process of directing problem-solving goals (Cohen et al., 1998) in a domain of discourse. Through mixed-initiative, the user and the formalism can share responsibility over domain goals. For example, Rich and Sidner (1997) and Rich and Sidner (1998) demonstrate a domain level collaboration through an interface agent that works on a plan with its user. Veloso (1996) and Veloso et al. (1997) employ a shared representation in the planning domain. Both automated and human planners are able to interact and construct plans jointly. Smith and Hipp (1994) propose a common meaning representation to achieve goals in natural language dialogue through mixed-initiative. Guinn (1996) considers initiative over mutually shared goals and how goals are solved by the participants (agent and human) in spoken dialogue systems.

Mixed-initiative over a domain goal requires both humans and automated software to share a representation of the domain of discourse. Designers share tasks with the formalism through an interaction model that connects the designer's view of the domain with the symbol level constructs available for computing exploration. From the

J.A. Jacko (Ed.): Human-Computer Interaction, Part I, HCII 2011, LNCS 6761, pp. 185–193, 2011.

designer's perspective, the representation of the domain must account for and connect onto the concepts underpinning the design space formalism. A difficulty of explanation arises in this task: the elements of the domain layer collapse into and find explanation in the sparse symbol-level machinery below. One formal device in the substrate serves several concepts in the domain layer. To address these difficulties, it is necessary to maintain three levels in the exposition of the domain layer: the designer's view of the components of exploration, the formal substrate underpinning these views and finally, domain layer concepts that map the user level concepts onto the formal components of the design space formalism.

2 Interaction with a Description Formalism

The description formalism described in Woodbury et al. (1999), provides a formal substrate for supporting the entities of exploration, state, move and structure. The symbol substrate for design space exploration is reported in Woodbury et al. (1999). It implements a formal mechanism for computing exploration in terms of types, features, descriptions and resolution algorithms (Burrow and Woodbury, 1999; Woodbury et al., 2000). The formalism is based on typed feature structures (Carpenter, 1992) and extensions (Burrow, 2003). In particular, the representational properties of information ordering, partiality, intensionality, structure sharing and satisfiability are addressed in the formalism. To define how designers may employ the entities of the substrate at the user level, a model of interaction, as shown in Figure 1 is necessary.

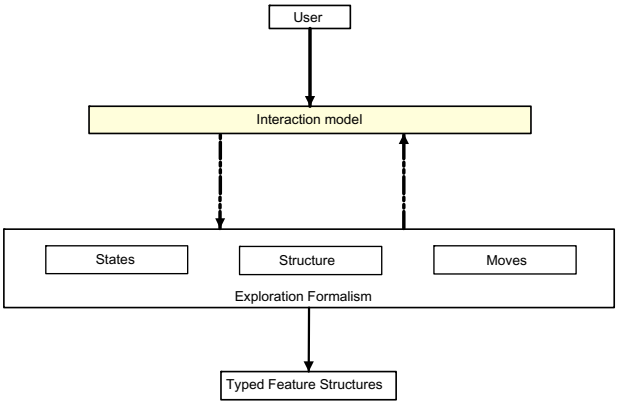

Fig. 1. An interaction model integrates the user and the description formalism

Mixed-initiative presents a paradigm, for combining a human designer and a description formalism through the specification of communication, coordination and control of the exploration process. To address the above, a three-layered model is developed for interactive exploration. Through the interaction model, both user and formalism must have the flexibility to acquire or relinquish initiative during exploration. The layers of the interaction model are shown in Figure 2. Each layer plays a distinct role for addressing the requirements for mixed-initiative exploration.

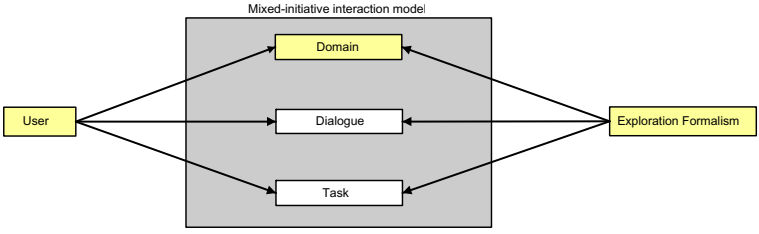

Fig. 2. A mixed-initiative interaction model integrates the user and the description formalism

The domain layer bridges the gap between the knowledge level formulation of the designer's view of the exploration domain and the symbol level substrate of the description formalism. The requirements of the domain layer in the mixed-initiative model of exploration are:

Support the designer's view of the domain. The domain layer must support the designer's view of the domain of exploration, namely, problems, solutions, choices and history. The domain layer provides concepts for the representation of problems, their reformulation and the generation of alternative solutions from the user's perspective. The domain layer mediates between the designer's view of exploration comprising problems, solutions, choices and history; and a design space representation aimed at efficient generation, indexing and recall.

Support joint responsibility over goals. Supporting joint responsibility over domain goals is a major problem in design exploration. Through mixed-initiative, the domain layer enables both the designer and the formalism to maintain context and share responsibility over goals in the domain of exploration.

3 Implementation

The designer's view of exploration comprises an account of problems, solutions, choices, their connections and the developing space of explicitly discovered design alternatives. This view is less concerned with the formal specification of internals and more with the existence of objects and the external hooks necessary to support interactive exploration. The designer's model of exploration comprises problems, solutions, choices and history (their connections and the resulting explicit design space). The problem formulation and reformulation cycle, the solution generation and reuse cycle, the intentional choices of the designer and the rationale of exploration in the form of a history are captured in this view. The representation of the designer's view is shown in Figure 3.

The structure of exploration is represented through the ordering relation of subsumption. The concept of an ordered design space underpins the description formalism. Exploration states are ordered by the relation of subsumption, a formal ordering over the collection of states. Burrow and Woodbury (2001) argue that design history is a significant device for supporting exploration. Choices, their connections

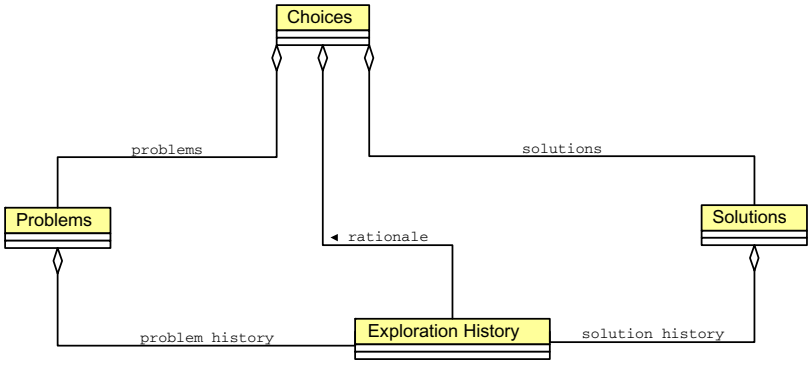

Fig. 3. The designer's view of exploration can be captured through a representation of the following entities: problems, solutions, choices and history. The problem formulation and reformulation cycle, the solution generation and reuse cycle, the intentional choices of the designer and the rationale of exploration in the form of a history are captured in this view.

and the developing history of explicitly discovered design alternatives must be accessible to the designer through interaction with the structure of exploration. The interaction model provides the designer with a view of design space structure. From the designer's perspective such a model must captures the elements of the history of design exploration.

The designer's view can be identified with the entities described in Section 2. It is necessary to explain these at a second level: that answers to both the designer's model and to the formal substrate. The mapping from the designer's view of exploration to the symbol structures of the underlying machinery is made explicit through four constructs in the domain layer, namely, problem state, solution state, choice and satisfier space. Figure 4 shows the mapping of the designer's view of exploration to the constructs in the domain layer.

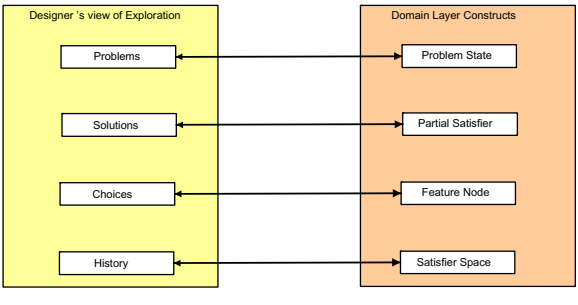

Fig. 4. Mapping the designer's view of exploration to constructs in the domain layer

Problems become problem states. Solutions become partial satisfiers. Choices compose a relation between problem states and partial satisfiers. Problems, solutions and choices recording the history of exploration are captured by satisfier space. Each of these constructs are developed in greater detail, following a three level exposition,

the designer's view, the symbol substrate, and the mapping of the two based on the domain layer as shown in Figure 4.

3.1 Problem State, Pstate

To a designer, problems comprise both design requirements and desired properties of an artifact. Problems may be hierarchical, that is in addition to requirements and properties they may comprise sub-problems, which themselves may be similarly recursive. Designers revise problems as aspects of a design situation reveal themselves through exploration, the conception of the actual problem being solved may change. In a designer's problem space, work is done by a combination of problem formulation (specifying/adding) requirements, attributes and sub-problems) and problem revision (removing or modifying the same).

At the symbol substrate, problems are specified through two main constructs of the typed feature structure mechanism: a type hierarchy **<Type, ≤ >** and a description drawn from Desc. First, the specification of a problem amounts to the construction of an inheritance hierarchy of types. The types are refined by a collection of features introducing appropriateness constraints on types. Second, a problem to be solved is expressed using descriptions drawn from Desc with respect to **<Type, ≤ >**. Present here is the representation of problems through type hierarchy construction and the authoring of descriptions. Detailed expositions of how these structures are supported in given in Woodbury et al. (1999).

As shown in Figure 5, the domain layer construct, Pstate, maps to the type hierarchy and to descriptions. From the perspective of an exploration process, a design problem can be expressed as a type inheritance hierarchy. Typically these would be such forms that have been ossified by past experience in design space. In this case, the problem would be available as a problem state, specified by its trivial description as a type in the type hierarchy. The mapping of a problem state to the type system can be explained as follows:

$$\text{PState} \equiv \bullet\text{PState} : \text{InheritanceHierarchy} : \text{Type} \tag{1}$$

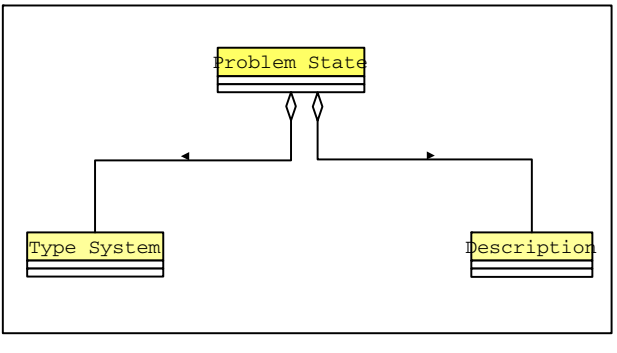

Fig. 5. The problem state composes a collection of descriptions, Desc over a type system

A problem can also be specified as a description specifying certain forms of spatial relations or constraints on types based on the description language. The connection between a PState and the formal substrate of descriptions drawn from Desc is given as follows:

$$\text{PState} \equiv \ \bullet\text{PState} : \text{Desc} : \text{Description} \tag{2}$$

In the above case, the exploration of the initial description would amount to designer interaction with the the domain layer construct, a PState. Through interaction with a PState, the designer can define design requirements either within the type system or through a collection of descriptions. Figure 6 extends Figure 5 by expanding a problem state to reveal its connections with typed feature structures. Summarising, the problem state is defined as a type or a description over a hierarchy of types. The domain layer construct Pstate, encapsulates the two distinct views of the problem formulation and reformulation process in the combined representation.

3.2 Solution States, SState

In the domain of design, problems may have no solutions, a finite number of solutions or an arbitrarily large collection of solutions. For example, in the SEED Knowledge Level, a solution is modelled as a "design_unit" (Flemming and Woodbury, 1995), representing a physical and geometric model. To a designer, a solution is a component in the spatial or physical structure of a building and has an identifiable spatial boundary. Thus solutions describe physical and geometric characteristics of structures satisfying problem descriptions.

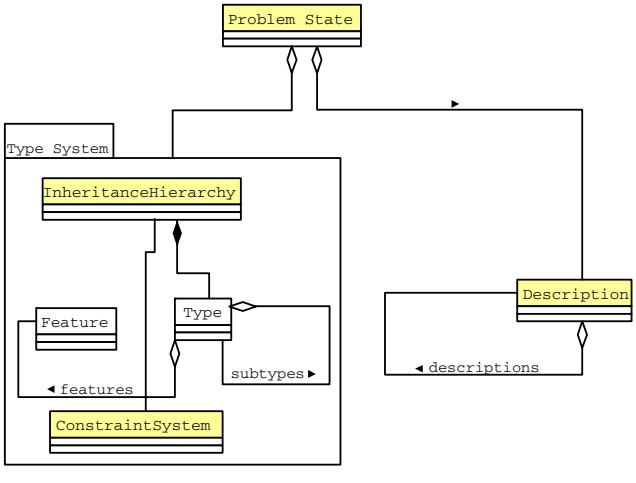

Fig. 6. Types, Features, constraints and descriptions comprise the representation layer for defining, decomposing and revising problems in the domain layer

The steps in the problem/design satisfaction relation are realised as incremental π-resolution states (Burrow and Woodbury, 1999) termed a partial satisfier and represented here as TFSPartialSatisfier. Partial Satisfiers represent initial,

intermediate (partial) and fully resolved solutions of a given description. A TFSPartialSatisfier composes descriptions and feature structures as follows:

$$\text{TFSPartialSatisfier} \equiv \bullet\text{PartialSatisfier: Desc: FeatureStructure} \quad (3)$$

The label PSat is used as a shorthand term for representing the relationship between satisfiers and descriptions in the substrate. A PSat composes a collection of TFSPartialSatisfiers:

$$\text{PSat} \equiv \bullet\text{TFSPartialSatisfiers} \quad (4)$$

In the domain layer, the view of a solution is encapsulated in solution state objects, written as SState. As an object, a solution state composes both resolved designs and partial satisfiers. The constituents of a solution state are shown in Figure 7. To a designer, a solution state provides a view on a developing design.

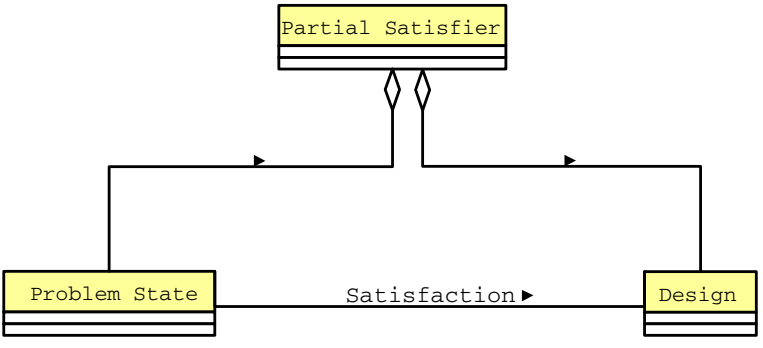

Fig. 7. Solution states compose partial designs with respect to an inheritance hierarchy of types and a partial satisfier

The realisation of the design as a typed feature structure and the satisfaction relation between problems and solutions as an incremental π-resolution state is shown in Figure 8. The solution state construct specifically adds intentionality concerning solutions to a problem specification. The connection of the solution state, SState to the formal substrate is given as follows:

$$\text{SState} \equiv \bullet\text{SState} \bullet \text{PState} : \text{Desc} : \text{TFSPartialSatisfier} \quad (5)$$

Summarising, the domain layer construct SState represents the notion of a design solution. In it, are embedded the symbol substrate concepts of description, the satisfaction of a description as satisfiers and the trace of intermediate solutions as partial satisfiers.

3.3 Choice

Problem formulation and generated solutions engender a large space of alternatives. These alternatives form a solution hierarchy and as designers revise solutions, a revision history of solutions is recorded. Support for the actions of the designer in

making choices about solution alternatives and solution revision is necessary. Choice records commitments and thus the relation amongst functional requirements and characteristics of a physical structure that satisfy these requirements.

The connections between a problem and its possible solutions (partial or complete) are encapsulated as choices. Choices represent both the intentional commitments made by the designer (selection) and the alternative paths of resolution uncovered by the formalism. The choices made by the designer and the resultant choice points arising out of formal resolution are recorded as feature nodes in the domain layer. The FNode records user choices with respect to problem alternatives, incremental generation and the selection of solution alternatives. The feature node, FNode captures the relationship between a problem state, PState and an alternative design that is a partial solution to the problem, SState.

Further, a FNode composes typed feature structures in the underlying formalism. The user and the formalism participate in a process of incremental generation of partial solutions of a problem statement. First, in the selection of a FNode from the a collection of possible solutions. Second, in the specification of the next step of resolution. The connection between a FNode and a SState is given in the path form as follows:

$$\text{FNode} \equiv \bullet\,\text{FNode} \bullet \text{SState} : \text{FeatureStructure} \tag{6}$$

The FNode enables the user to make choices at a particular point in the problem formulation and solution generation process.

4 Conclusion

The paper presents an interaction model for supporting mixed-initiative in design exploration. It identifies the requirements of the domain layer and constructs a designer's view of exploration comprising problems, solutions, choices and history over the symbol level representation of design space exploration. The mapping is developed through the development of the concepts of: Problem state, Solution state, Choice and Satisfier space. In each of these constructs, the case for mixed-initiative is made through a three level exposition, the designer's view, the symbol system view and the mapping of the two through the domain layer constructs, PState, SState, and Choice. Problem states represent design problems. Solution states represent partial design solutions. Finally, the mapping of the domain layer to an interface construct, the FNode is demonstrated. Feature Nodes compose problem formulation and solution generation processes and support the exploration operations.

References

1. Burrow, A., Woodbury, R.: Design Spaces–The forgotten Artefact. In: Burry, M., Dawson, T., Rollo, J., Datta, S. (eds.) Hand, Eye, Mind, Digital, Mathematics and Design 2001, pp. 56–62. Deakin University, Australia (2001)
2. Burrow, A.: Computational Design and Formal Languages. Unpublished PhD Thesis, Department of Computer Science, The University of Adelaide (2003)

3. Burrow, A., Woodbury, R.: Π-resolution and Design Space Exploration. In: Augenbroe, G., Eastman, C. (eds.) Computers in Building: Proceedings of the CAADF 1999 Conference, pp. 291–308. Kluwer Academic Publishers, Dordrecht (1999)
4. Carpenter, B.: The Logic of Typed Feature Structures with applications to unification grammars, logic prorgams and constraint resolution. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge (1992)
5. Chien, S., Flemming, U.: Design Space Navigation: An Annotated Bibliography, Technical Report No EDRC 48-37-96, Engineering Design Research Center, Carnegie-Mellon University, Pittsburgh, PA, USA (1996)
6. Chien, S., Flemming, U.: Information Navigation in Generative Design Systems. In: Liu, Y., Tsou, J., Hou, J. (eds.) CAADRIA 1997, vol. 2, pp. 355–366. National Chia Tung University, Hsinchu (1997)
7. Cohen, R., Allaby, C., Cumbaa, C., Fitzgerald, M., Ho, K., Hui, B., Latulipe, C., Lu, F., Moussa, N., Pooley, D., Qian, A., Siddiqi, S.: What is Initiative? User Modeling and User-Adapted Interaction 8(3-4), 171–214 (1998)
8. Flemming, U., Woodbury, R.: Software Environment to Support Early Phases of Building Design (SEED): Overview. Journal of Architectural Engineering, ASCE 1(4), 147–153 (1995)
9. Guinn, C.I.: Mechanisms for Mixed-Initiative Human-Computer Collaborative Discourse. In: Joshi, A., Palmer, M. (eds.) Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics, pp. 278–285. Morgan Kaufmann Publishers, San Francisco (1996)
10. Rich, C., Sidner, C.L.: COLLAGEN: When Agents Collaborate with People. In: Johnson, W.L., Hayes-Roth, B. (eds.) Proceedings of the First International Conference on Autonomous Agents (Agents 1997), pp. 284–291. ACM Press, New York (1997)
11. Rich, C., Sidner, C.L.: COLLAGEN: A Collaboration Manager for Software Interface Agents. User Modeling and User-Adapted Interaction 8(3-4), 315–350 (1998)
12. Smith, R., Hipp, R.D.: Spoken Natural Language Dialog Systems: A Practical Approach. Oxford University Press, Oxford (1994)
13. Veloso, M.: Towards mixed-initiative rationale-supported planning. In: Tate, A. (ed.) Advanced Planning Technology, pp. 277–282. AAAI Press, Menlo Park (1996)
14. Veloso, M., Mulvehill, A., Cox, M.T.: Rationale-Supported Mixed-Initiative Case-Based Planning. In: Innovative Applications of Artificial Intelligence, Providence, RI, pp. 1072–1077 (1997)
15. Woodbury, R., Burrow, A., Datta, S., Chang, T.W.: Typed Feature Structures and Design Space Exploration. Artificial Intelligence in Design, Engineering and Manufacturing 13(4), 287–302 (1999), Special Issue on Generative Design Systems
16. Woodbury, R., Datta, S., Burrow, A.: Erasure in Design Space Exploration. In: Gero, J. (ed.) Artificial Intelligence in Design 2000, pp. 521–544. Kluwer Academic Publishers, Dordrecht (2000)