Flexible and distributed coordination models for collaborative design

CUMMING Michael

Delft University of Technology, Faculty of Architecture, Netherlands.
http://www.bk.tudelft.nl/users/cumming/

Abstract: Designers working in collaborative design situations, attempt to plan or anticipate their activities, such that their work progresses in an orderly manner, according to technical demands of their domain. Designers, and the organizations that employ them, often attempt to formally represent such plans using process representations, such critical path diagrams, or Petri nets. Such process articulation and formalization can have benefits for designers and organizations, such as standardization and improvement of work practices, and improved collaboration and coordination between design parties.

In addition to plan making, designers also try to coordinate their actions with the actions of others on the design team. This coordination, which often takes place in real time, is a process that is necessarily social, interactive, and iterative. Here the formulation of suitable process representations is more difficult, due to the dynamic and complex nature of social interactions. How to represent and design such coordination processes, is a continuing research question in the process modeling community. It is possible there exists general coordination mechanisms that could be useful in a variety of domains.

Possibilities for distributed methods of design process coordination are examined. A coordination method is proposed that involves the exchange of design process models, represented as Petri nets. Rather than concentrating on the specific content of these models - which is assumed to vary considerably between design domains - general coordinating mechanisms are proposed. One such mechanism involves the communication of social commitments to process models, in addition to communication of the content and authorship of these models.

***Keywords**: Collaboration, coordination, process modeling, Petri nets*

Introduction

Collaborative design can be a complex activity. It depends on the successful interaction of many different parties, sometimes with profoundly different perspectives on the design process, and design product. This research attempt to tame such complexity in a way that does not harm the collaborative design processes.

This research attempts to combine lessons derived from research from two more or less opposing camps: one, a community which values descriptive symbolic process modeling, and second, bottom-up construction of models involving social feedback, inspired by studies in situated cognition and complex interactive systems that display emergent properties and behaviors.

Background

'Plans-as-programs' vs. 'plans-as-communications'

Agre and Chapman in their influential report 'What are plans for? 'distinguish between two uses for symbolic plans: one as 'plans-as-programs', and 'plans-as-communications' (Agre & Chapman, 1989). Plans-as-programs are intended as algorithmic accounts of the steps to be followed in order to fulfill certain goals. This approach is consistent with the theories of an influential faction in cognitive science, which posits that following symbolic plans is necessary to enable rational, goal-directed activity. Plans-as-communication as seen as much more informal, linguistically based accounts of what should be done, and suggests that complex activity depends both on symbolic descriptive models, as well as real-time, situated improvisation. For the symbolic-modeling side of this vigorously debated issue see (Vera & Simon, 1993) while for the opposing, situated view see (Clancey, 1993).

According to Agre and Chapman, the plans-as-program approach suffers from the following problems: 1) it poses computationally intractable problems, 2) it is inadequate for a world characterized by unpredictable events such as the actions of other agents, 3) it requires that plans be too detailed, 4) it fails to address the problem of relating the plan text to the concrete situation (Agre & Chapman, 1989).

Plans-as-programs are seen as being insufficiently flexible, in that appropriate real world activity depends on real-time adaptive responses to uncertainties. These uncertainties are difficult or impossible to predict in advance. How to deal with uncertainties and contingencies is especially important in the field of robotics. A central problem within robotics is how to design robots capable of negotiating real-world environments. One approach to this problem is the work of Brooks, in which non-deliberative, reactive interactions between robots and the environmental stimuli they might encounter is favored, instead of plan construction and plan following (Brooks, 1991).

In agreement with the position of Brooks, Agre and Chapman view the problem with the plan-as-program view is that it understands activity as a matter of problem solving and control, rather than one that involves fashioning real-time adaptive responses to constantly changing situations. Similar issues are relevant in the domain of design process support, in which adaptive, 'opportunistic' responses to complex, dynamic situations are also important. Yet, Agre and Chapman do not see plans-as-programs as necessarily ineffectual: in situations that have relatively static, well-defined semantics, and low levels of uncertainty, they can be very useful.

In contrast, the plan-as-communication approach sees the plans as resources, among many other resources, which agents may choose to use or not to use, in the context of complex activities. Additional resources could be many things such as the opinions of others, clues from the environment, the contents of other plans, etc.

Here, the contents of a plan are not directly connected to a cognitive, perceptual, and motor system that depends on access to detailed plans in order to function at all – which is the case with plan-directed robots. Therefore, items in a plan have a much less central role to play than in the plans-as-program view, where plan contents are used not only to inform, but also to structure and to effect activity. The plans-as-communications approach assumes the existence of a general cognitive ability beyond that of plan construction and execution, and depends on an agent understanding the meaning of an item on a plan.

Plans-as-programs are useful in agents that inhabit simple, static environments in which the execution of relatively static symbolic plans is suitable. This assumes that there exists some agent who is capable of creating the plan in the first place, and that once created, the plan will not have to be replanned at every step of its execution, due to unforeseen contingencies.

These criticisms are relevant to collaborative design where there is usually no one party that is capable of devising a plan that will allow all the activities of a design team to be structured. In addition, even if a plan did exist it is unclear how closely a design team would want or be capable of following it. Finally, due to the unpredictable nature of collaborative design, it is clear that any plan will have to be under constant revision, in order to cope with the changing circumstances, re-interpretations of requirements, and design opportunism common within design teams. Therefore, the plans-as-communications approach appears to be a more realistic model of how plans are actually used in collaborative design.

Introduction to collaborative design

Collaborative design is a common way of designing, yet it tends to be a very complex activity. Collaborative design depends on the successful interaction of many different parties. The nature and outcome of these interactions can be quite ad hoc, and specific in nature, and therefore difficult to predict, and to generalize. Design problems are also becoming more complex, with increasing integration demanded between diverse and possibly novel functional requirements.

Increased complexity in design has both social and technical aspects. Not only are the technical problems becoming more difficult, such as learning to work with new materials, or learning to cope with changing regulatory environments, but the social demands that they bring is also changing. People from different cultures, who may have never worked together before, are brought together and expected to be quickly bridge striking cultural differences and become productive with one another.

Models of collaborative design

Collaborative design can be studied both from a top-down, and bottom-up perspective. The top-down perspective focuses on global design issues such as how well a completed artifact fulfills its primary design requirements, or how much it costs. The bottom-up approach focuses on interactions between low-level entities such as individual design requirements, processes, or designers. In both perspectives what constitutes the ‘top’ or ‘bottom’ requires definition by an interested observer, rather then being an objectively established, invariant feature of the process.

From a bottom-up perspective, collaborative design can be modeled as a complex system. Complex systems research addresses at a fundamental level, the behaviors of interdependent entities. Complex systems typically have no central controller, and the global behaviors they exhibit, emerge because of local concurrent actions (Klein, Sayama, Faratin, & Bar-Yam, 2001). Biological systems, such as ecosystems and organisms, are perhaps the most commonly presented examples of complex systems (Resnick, 1994).

According to Klein (2001), designers, as well as design issues, can be modeled as ‘nodes’ in dependency networks. In such a view, completing a collaborative design process, involves designers attempting to maximize the value of a (hypothetical) global utility function. Klein notes that the problem with collaborative design in general, is that the networks that most realistically model how collaborative design is done in practice, and ought to be done in practice, are also the ones that display the most complicated behaviors.

In collaborative design, this situation means that incremental improvements to a given design configuration, such as product models as they currently appear, may improve the designs, but will not necessarily lead to global optima.

Within the Distributed Artificial Intelligence (DAI) community, the strategy of distributing control, data, as well as knowledge sources, is now widely supported (Whitfield, Coates, Duffy, & Hills, 2000). Such an approach has been shown to have several advantages, including the reduction of performance bottlenecks, the increase in reliability, and the soft, rather than steep or complete degradation of performance, when systems are under stress.

Distributing control and data can also have disadvantages according to Jennings, in that 1) each agent only has a partial and imprecise perspective, 2) there is increased uncertainty about each agent's actions, 3) it is more difficult to attain global behavior, and 4) the dynamics of such systems become extremely complex.

However, distributed control when placed in a design context is not a concept that may not have much intuitive appeal to designers. Designers are usually trained to view their primary job description as controllers of design processes. Complex collaborative design processes can indeed be centrally controlled. However, once design projects get to a certain degree of complexity, central control can become a bottleneck that degrades the performance of the whole system. Two common strategies to avoid such degradation are: 1) by not attempting to explore non-routine solutions, or 2), by choosing to downplay the importance of some of the design requirements. In general, both these strategies appear to be less than ideal.

If centralized control does not scale well, then forms of distributed control become necessary. This means of course that the final product may not be guided by the vision of a single individual. However, it does means that designs may become better coordinated, and provide higher quality solutions to complex multi-dimensional problems.

The trend towards integrated product models must also be noted. Since it is usually a single unified artifact that is the intended result of a collaborative design process, it seems to make sense to attempt to make unified design product representations from the beginning stages of design. Unified product models in which all the design description information resides in one location, can be, for instance, very convenient when checking for completeness and consistency (Flemming & Woodbury, 1995).

Coordination science

Coordination science is a new discipline that has been developed to help explain and manage complex collaborative situations, which tend to overwhelm existing process management theory and technique. See (Whitfield et al., 2000), (Klein, 1998), and (Malone & Crowston, 1992) for overviews. Coordination of action is required, according to Klein (1998), when distributed activities, such as that found in collaborative design, are interdependent.

A good, concise definition of coordination is that provided in (Malone & Crowston, 1992) 'the act of working together harmoniously'. Malone and Crowston also provide a list of technical definitions others have proposed for the term.

According to Klein (1998), the most fundamental aspect of support for coordination comes through communication. That is, it is inconceivable that in whatever design coordination regime, whether software-based or otherwise, that agents will be able to coordinate their work without actually communicating with one another.

Jennings proposes that coordination is built upon four main structures: commitments, conventions, social conventions, and local reasoning capabilities.

If an agent commits itself to perform a particular action, then, provided that circumstances do not change, it will endeavor to honor that pledge (Jennings, 1996). Non-performance of a commitment, made in a social setting, can entail social costs, which people sometimes go to extraordinary lengths to avoid. From a distributed systems perspective, commitment by agents to a course of action adds a degree of certainty, to future events. Jennings stresses the importance of social commitment with the following hypothesis:

Centrality of Commitments and Conventions Hypothesis

- All coordination mechanisms can ultimately reduced to commitments and their associated (social) conventions,

- commitments are viewed as pledges to undertake a specified course of action, and
- conventions provide a means of monitoring commitments in changing circumstances (Jennings, 1996)

Design of an application

The intention behind this research is not just to create theory, but also to construct a software prototype that demonstrates ideas regarding distributed process coordination and emergence. This application is currently in implementation.

Application features

The intention here is to provide resources for designers in the form of symbolic process models, yet allow this approach to be informed by developments in situated cognition, and in complex interactive systems. The approach for encouraging this to happen is involves:

- viewing process models as communications that inform a design process, rather than control it.
- by not assuming that all knowledge needed to manage design processes necessarily comes from a central expert or authority. Rather, consider ideas that might be useful in influencing a design process, could conceivably come from a variety of distributed sources.
- by allowing a multiplicity of process models to be developed, which form 'populations' of models, rather than attempting to unify these models into a single logically consistent entities.
- by providing mechanisms for filtering and ranking the contributions from multiple sources, such that some order can be derived from these populations.
- by not imposing process models externally onto a design process, but rather by allowing designers working within the process to define models for themselves. Therefore, the intention is to converge the class of authors of process models with that of users of such models.

The application uses simple message-based peer-to-peer communication, in which there is no central controller. Individual nodes on the network serve both as clients – consumers of information, and servers – sources of information. Nodes on the network represent one designer, or other participant in a design process.

- A user can do three main things using the application: he can send messages to others, send messages to himself, or receive messages from others.
- Whom users can send messages to is not prescribed by the application – messages can be sent to anyone else who also has the application.
- The application contains no specific process content itself, rather it supports the coordination of content that users add to it.
- The way it supports coordination, is by representing the level of commitment that users have with respect to the planning and performance of tasks.
- The tasks to be completed in a collaborative design project comprise the process content that users add themselves. These individual tasks when viewed together are simple 'to-do' lists. The to-do list is viewed as one of the simplest, yet most effective process models. Despite their simplicity, they can be very useful, especially if their content is known to be relevant, and appropriate to a current situation.

Tasks are seen as social objects that require construction within a social environment to acquire their relevance and significance. There are three participants directly involved in this process: Author: the person who first comes up with the idea that a task should be performed, and then communicates this idea to others, Performer: the person who performs a task, and Customer: the person for whom a task is being performed. A User, referred to below, is anyone using the application.

An 'Author' category has also been added. This acknowledges that ideas regarding what to do in collaborative design can sometimes come from parties who may have no role in either performing or being Customers to these tasks.

From the perspective of one user, tasks have five states

- New: the task is new to the user, so has not yet been viewed,
- In-Pre-Negotiation: the interested parties decide whether such a task should be performed,
- In-Performance: the Performer is to perform the task, and to communicate to others when this performance is thought to be complete,
- In-Post-Negotiation: the Performer and the Customer decide whether the task has been performed to everyone's satisfaction, and
- Retired: the task is no longer active. Retired tasks may be of interest for historical reasons such as for purposes of recording design process histories or design rationale. Retired tasks can also be reused and re-communicated and can therefore become New again.

User actions

First-Contact (New to In-Pre-Negotiation): a user views a task for the first time.

- Agree-To-Perform (In-Pre-Negotiation to In-Performance): both the Performer and Customer agree that the task should be performed as described.

Complete-Performance (In Performance to In-Post-Negotiation): the Performer informs the Customer that the task has been completed.

- Agree-To-Retire (In-Post-Negotiation to Retired): the Customer informs the Performer that the results of the task, as performed, are acceptable.

Reuse (Retired to New): a user reuses an existing Retired task, by communicating it to someone.

- The above state and transition information can be represented as a simple Petri net.

Messages communicated

Three types of messages are communicated:
Task messages: these are the actual process content. Each task message has four types of attributes: 1) Description of the task, 2) People involved in the task: Author, Customer, and Performers of the task, 3) Time frame: 'AgreeByDate', 'PerformByDate', and 4) Task state.

Input messages: this information determines whether a specific task message can change its state. Task messages that do not acquire corresponding input messages, do not advance and can be stranded in a particular state forever. Tasks, which do not move for a specified period, are garbage collected. Each input message has three types of attributes:

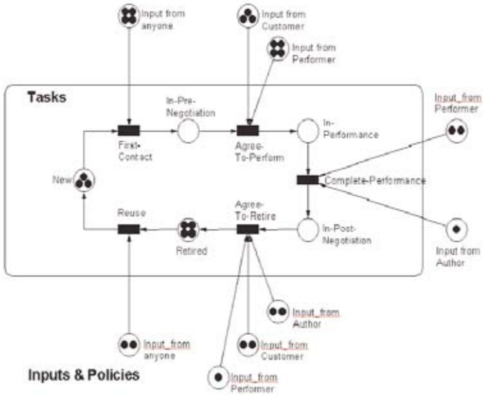

Figure 1. Petri net showing tasks, states, and task inputs. Diagram output from the Petri net application Visual Object Net ++ (Drath, 2000)

1) the user who is the required source for the input, 2) Task to which it refers, and 3) the task state for which the input is relevant.

Policy messages: this information specifies the parties whose inputs are required for a Task to advance to its next state. These too are specific for particular Tasks at a certain state, and they contain a list Users whose input is required.

Certain transitions require input fraom certain users as shown by Fig. X above. For instance, in order for a task to change its state from In-Pre-Negotiation to In-Performance, inputs tokens from both Performers, and Customers are required. This requirement is a rule that is defined graphically by the configuration of places and their transitions in the Petri net. In Petri nets, tokens must be present in all input places for a transition to be enabled (Jensen, 1996).

This rule – that tasks with the state of In-Pre-Negotiation can only progress with the input of both Performers and Customers – is a type of workflow policy. This policy is not a general one, but could change for different organizations, design projects, or tasks.

In a distributed environment important questions become: who defines what appropriate policies should be, and where does this policy information reside?

Policies are another type of message. They, like other messages, have an author, specify a task and a task state, and list the parties for which input is required. Therefore, in this application anyone can define a policy. Whether it turns out to be a sensible, reusable policy is up for the group to decide.

Filtering a population of plans

As tasks journey from New to Retired and possibly around again, each step requires user intervention, interactions, and later, agreements from involved parties. Such intervention and agreement may be difficult or impossible to acquire for any one task. Therefore, the state transitions represent a kind of social filtering mechanism. Here, tasks at the beginning encounter little or no filtering, while those later in the state transition system, encounter much more. Such activity is viewed as a type of commitment-constructing exercise between distributed users to perform certain tasks.

Those tasks that do become Retired tend to reflect those activities that people within collaborative design projects find useful and to which they are willing to commit. Since social involvement and commitment are required to attain such a status, this suggests that such well-filtered tasks are also ones more likely to be reused by a similar social group in the future. If one sees organizations as 'networks of recurrent conversations', as does Winograd (Winograd & Flores, 1987) p. 158, then this filtering mechanism has the effect of separating tasks that are likely to become recurrent, from those that are not.

In addition to indicating the degree of social commitment, or probability of recurrency, additional semantics might be able to be inferred from tasks that make it to certain states but no further. For instance, if a task remains forever In-Performance, with the Performer never completing it, this might suggest that this task is too difficult to perform. If a task remains In-Post-Negotiation, thereby indicating that the Performer believes she has completed it, this may suggest that the Customer for that task is not satisfied with the Performer's work.

Implementation

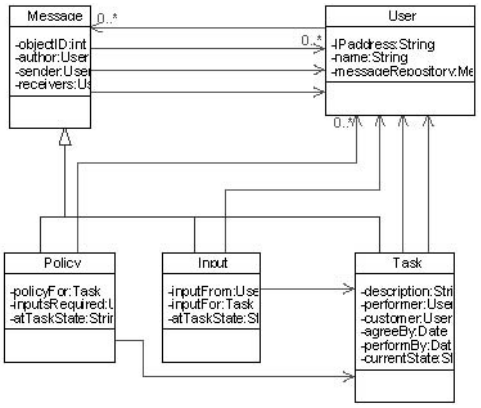

Figure 2. UML class diagram of Messages and User

The process content is represented as Petri nets, communicated in a lightweight fashion using Petri net/XML technologies [Aarhus, 2000 #503]. Petri nets are a well known process representation with several compelling advantages:

- A clear graphical representation,
- the ability to handle both state and task process information equally well,
- a syntax and semantics based on a small number of simple ideas,
- the ability to execute models dynamically, and
- the ability to model true concurrency correctly (Jensen, 1996).

Such advantages make them particularly well suited to the modeling, for instance, of distributed algorithms (Reisig, 1998).

Conclusion

Design is often thought of as process that requires a single individual in a position of authority to insure that the final design has a consistent conceptual integrity. This strategy works well for small design projects, but becomes difficult for larger ones, which may have so many requirements, and aspects that it becomes impossible for single individuals to understand the issues, keep track, and maintain control. This research proposes a distributed bottom-up method of design coordination. The involving peer-to-peer sharing of simple process models in which levels of social commitment are represented. If such commitment sharing mechanisms are iterated in a social context, it is proposed that the processes of complex collaborative design projects can be become better coordinated.

References

Agre, P. and Chapman, D.: 1989, What are plans for? A. I. Memo 1050a, Artificial Intelligence Laboratory, MIT, Cambridge, MA.

Brooks, R.: 1991, Intelligence without Representation, Artificial Intelligence, 47, pp. 139-160.

Clancey, W.: 1993, Situated Action: A Neuropsychological Interpretation Response to Vera and Simon, Cognitive Science No. 17), pp. 87-116.

Drath, R.: 2000, A short user guide for Visual Object Net++, ver. 1.44, Rainer Drath. Available: http://www.R-Drath.de/VON/von_e.htm.

Evan, W.: 1993, Organizational Theory: Research and Design, Macmillan, New York, NY.

Flemming, U. and Woodbury, R.: 1995, Software Environment to Support Early Phases in Building Design (SEED): Overview, Journal of Architectural Engineering, 1 (4), pp. 147-152.

Jennings, N. R.: 1996, Coordination techniques for distributed artificial intelligence, in G. O'Hare and N. R. Jennings (eds.), Foundations of Distributed Artificial Intelligence, pp. 187-210, John Wiley & Sons, New York, NY.

Jensen, K.: 1996, Coloured Petri Nets: Basic Concepts. 2nd ed., Springer Verlag, Berlin.

Klein, M.: 1998, Coordination Science: Challenges and Directions, in W. Conen and G. Neumann (eds.), Coordination Technology for Collaborative Applications, pp. 161-176, Springer Verlag, Berlin.

Klein, M., Sayama, H., Faratin, P. and Bar-Yam, Y.: 2001, What Complex Systems Research Can Teach Us About Collaborative Design, Proceedings of International Workshop on CSCW in Design, London, Ontario, Canada, pp. 5-12.

Malone, T. and Crowston, K.: 1992, What is Coordination Theory and How Can It Help Design Cooperative Work Systems?, in D. Marca and G. Bock (eds.), Groupware: Software for Computer-Supported Cooperative Work, pp., IEEE Computer Society Press, Los Alamitos, CA.

Reisig, W.: 1998, Elements of Distributed Algorithms: Modeling and Analysis with Petri Nets, Springer Verlag, Berlin.

Resnick, M.: 1994, Turtles, Termites, and Traffic Jams: Explorations in massively parallel microworlds, The MIT Press, Cambridge, MA.

Vera, A. and Simon, H.: 1993, Situated Action: A Symbolic Interpretation, Cognitive Science, 17, pp. 7-48.

Whitfield, R. I., Coates, G., Duffy, A. and Hills, B.: 2000, Coordination Approaches and Systems – Part I: A Strategic Perspective, Research in Engineering Design, 12, pp. 48-60.

Winograd, T. and Flores, F.: 1987, Understanding Computers and Cognition: A new foundation for design, Addison-Wesley, Reading, MA.