From: AAAI Technical Report SS-94-07. Compilation copyright © 1994, AAAI (www.aaai.org). All rights reserved.

Modeling Organizations: Methods Used by Real-world Designers

Submission to the AAAI-94 Spring Symposium on
Computation Organization Design

Bryan Borys
Computational Organizational Design Lab
Institute for Safety and System Management
University of Southern California
Los Angeles, CA 90089-0021
213/740-4045
borys@usc.edu

Formal, abstract computational techniques for modeling organizations imply certain approaches to certain modeling problems. Less formal techniques used by people involved in concrete design contexts in real organizations (real-world designers) face the same problems, but use different approaches. Four such problems are considered: the problems of simplification (which features to model, which to ignore), granularity (what unit of analysis to adopt), simulation (how to verify the accuracy and consistency of the model), and agency (how to be certain that the agents modeled will respond to certain signals in certain ways). Approaches of computational organization design (COD) and real-world design (RWD) to each of these problems are compared and some lessons for both COD and RWD are drawn.

Introduction

Central problems of computation organization design (COD) result from the gap between, on one hand, "natural" representations of organizations that are meaningful to organizational members, and on the other hand the formal representations required by available computational techniques. COD proceeds by making simplifying assumptions about organizations which make computation easier. COD modelers sacrifice faithfulness to reality in the name of modeling expediency, often with the hope that further refinements will yield more faithful models.

Analogous problems face designers of organizations in "real-world" settings.[1] Practitioners of real-world design (RWD) must also transform their chaotic, ambiguous and confusing observations of organizational behavior into a format amenable to design.[2]

[1] I use quotes around the term "real-world" (in this instance only) in recognition of the fact that the distinction between real-world design and COD is one of degree: "real-world" designers use formal techniques just as do those involved in creating or using COD systems.
[2] For detailed discussions of this problem, see Bryan Borys, *Where Do Rules Come From?: Participant-Observation of the Process of Administrative Innovation,* unpublished doctoral dissertation, Graduate School of

Nonetheless, the activities of real-world designers are more deeply embedded in a particular organizational context than that of COD practitioners. The task of COD is to generate an *abstract* representation of organizational behavior amenable to simulation, generation of design alternatives, etc. The task of RWD is to generate a *concrete,* workable design that will address specific problems in a specific organization.

In pursing RWD, designers have access to, and are influenced by, a rich set of organization-specific knowledge (e.g., culture, standard practice, knowledge of individual idiosyncrasies). What might be considered extraneous issues from a COD perspective are kept at the forefront of the RWD task. For instance, some real-world designers tend to ignore certain organizational features that might embarrass or offend.[3] Thus real-world design is more deeply *embedded* in the rich, complex and often unprogrammable world that sociologists with a penchant for neologism call "everyday life."

This embeddedness has two implications that differentiate RWD from COD: First, since RWD is embedded in an organizational context, real-world designers have access to knowledge that helps them get around tricky modeling problems— they thus have a leg up on the more abstract and formalistic COD designers. Second, however, this same access sometimes leads real-world designers to generate models that are more influenced by the context of the design task than by its explicit requirements — they are thus more vulnerable to bias than the COD practitioner pursuing abstract models.

For real-world designers, bridging the gap between organizations and models of organizations involves applying two things: First, their native understandings of the organization they are attempting to (re)design. Second, the everyday methods they use to manage the rich complexity of everyday life. The approaches they use for bridging the gap and managing the problems of organizational modeling provide a fruitful counterpoint to more formal and abstract approaches developed in the COD literature.

Real-world design: Central problems and approaches

What follows is a set of observations of a group of real-world organization designers. These designers were members of a formal process improvement team at a large electronics manufacturer I shall call "Electra."[4] The Electra design team was charged with using basic "business process reengineering" methods[5] to improve the speed of Electra's engineering change process. The most difficult part of this task was to map the existing engineering change process; this step took up about two-thirds of the design team's time.

I shall highlight some of the major problems that faced these real-world designers and identify

Business, Stanford University, 1992; and "'Seeing the Elephant': A Grounded Theory of the Role of Knowledge Interests in Business Process Reengineering," unpublished manuscript, submitted to the 1994 Academy of Management meetings.

[3] See Borys, 1992, op cit.

[4] More detailed descriptions and analyses of the Electra case can be found in Borys, 1992, op cit.

[5] See, for instance, Hammer, Michael and Champy, James. 1993. *Reengineering the Corporation: A Manifesto for Business Revolution*, New York: HarperBusiness.

how the designers approached the problems. From this discussion I shall draw some lessons, or at least some cautions, for those working in COD.

1) The problem of simplification.

All models simplify. COD models often simplify on the basis of computational ease, making assumptions that are necessary to allow the use of available methods and representations.

Real-world designers simplify, as well. The representational tools available to real-world designers can be quite rich: flow charts, diagrams, organization charts and prose descriptions of activities can convey a richness of meaning and a level of complexity that is infeasible for more formal representations. Nonetheless, these tools impose limits on complexity, as well. Thus real-world designers also face the problem of when and how to simplify, to gloss, to ignore.

The qualitative difference between real-world design and COD is that real-world designers often simplify on the basis of "practical" demands — their simplifications facilitate successful action in a specific organizational context, not merely successful computation or accurate representation. Real-world designs are often evaluated in terms of their implications for action — even to the point where implementability is a greater concern than accuracy in representation. For instance, designers tend to model features that create important and fixable problems and ignore modelable features that generate unimportant problems or problems whose solutions lie beyond the capacities of the designers.[6]

Thus the implementation context (rather than merely the available evidence and modeling tools) influences how the model is simplified. One result is that real-world designs tend to more readily implementable than COD designs, because implementation issues are (often surreptitiously) imported into the modeling process. The other side of this coin, however, is that RWD may merely replicate existing organizational problems, where more formal and abstract COD techniques force designers to address them.

This suggests that design-for-implementability (DFI) is an important consideration: COD designs, generated by a COD expert far from actual organizations, may ignore crucial factors that impede the utility of the resulting design. It also highlights, however, the danger that DFI may serve merely to legitimize the glossing of important organizational problems.

2) The problem of granularity.

In modeling behavioral processes, an important issue is to choose the appropriate unit of analysis — that is, should we model behaviors, tasks, inputs/outputs, "steps" in a process, etc.? COD models typically adopt some stylized unit (e.g., "transaction") that is hard-coded into the formal representational scheme. COD designs thus tend to consistently implement a particular choice of granularity throughout the model.

---

[6] See Borys, 1992, op cit.

For real-world designers, determining the appropriate granularity is a constant problem. What is appropriate is determined in part by more or less "natural" chunks of activities. The level of granularity may differ across parts of the organization that is being modeled. For instance, activities within a Finance function are often governed by quite explicit rules governing quite "small" tasks. These provide a language for rendering Finance-related activities in a fine-grained way; this language is adoptable by real-world designers. The language of Design Engineering, by contrast, is full of "black boxes," in which many activities are implied, but not analyzed (e.g., "generate design alternatives"). Thus readily available models of Design Engineering tend to be coarser-grained than those of Finance. A task in Finance may take one person several minutes; yet a task in Design Engineering may take several people several months. The language of tasks in Finance and in Design Engineering facilitate modeling at quite different levels of detail.

Organizational politics seem to play a role in such distinctions. Given readily available models of Design Engineering, it is nonetheless possible to delve into the design task in more detail (e.g., by asking "How do you generate design alternatives?"). Design engineers are often resistant to such probing, however. Their defense is that their work is not amenable to such fine-grained analysis. As I tried to uncover the fine detail of a particular design engineering decision, for instance, two engineers in separate conversations gave the same response: They could not explain the steps involved in that particular process; all they could say was that "it's an engineering decision." They implied that an outsider could not comprehend it in greater detail. Such "black boxing" of certain areas protects them from outsiders' meddling.

It also, however, influences the granularity of real-world designers' models. Thus the choice of how fine-grained a model should be is influenced, for real-world designers, as much by language, organizational structure, and organizational politics as by the formal requirements of the modeling environment. Again we find a two-sided result: RWD sacrifices consistent implementation and is more open to bias than more formal methods. But on the other hand it more easily captures models of task structures that are more inherently meaningful than some COD ontologies.

3) The problem of simulation.

A major benefit of COD is that it facilitates simulation of various organizational features. Such simulation allows the COD modeler to generate a result, compare it to empirical results, and thus verify the accuracy and consistency of the model.

For real-world designers, simulation and consistency-checking is both more rich and less systematic and accurate. Since real-world designs are typically lists of features or steps in a paper flow chart, real-world designers must supply their own medium of simulation and test. The most natural and perhaps the most pervasive is the narrative: real-world designers "talk through" the implications of their models, checking whether the models "make sense."

The problem for real-world design is that forcing models to "make sense" in the context adopted by the real-world design group may introduce spurious elements. "Narrative

simulation" forces the elements of the model into a specific context in which they may not make sense. Real-world designers thus may sacrifice accuracy for meaningfulness.

One by-product of narrative simulation is "filling in the gaps": a frequent source of spurious information. Narratives encourage the narrator to introduce his or her own knowledge in the place of information that missing from the story. For instance, Electra designers often filled in the wholes in informants' accounts of the engineering change process. Once they had mapped several pieces of the process, they talked through an example engineering change. At several points they found gaps in the model. The typically response was to say "what they must mean is..." and to fill that gap with their own (often wrong) knowledge.

Another by-product is that elements that appear logically inconsistent are often removed or glossed, rather than accepted as evidence of the designers' lack of understanding.[7] Narratives make sense only within a particular context. For instance, simulating the engineering change process model by imagining the history of an engineering change requiring a change to a printed circuit board layout might generate different issues than a narrative history of a fictitious change involving a new electronic component.

The result is that simulation in real-world design, insofar as it relies on placing a particular model within a specific context, limits the robustness of the simulation. The flip side of this is that, again, the end result is a model that has greater a priori meaningfulness. For COD, simulation and checking increases the internal consistency from a very formal standpoint. For real-world design, the process often cannot provide the same level of rigor. Instead, however, real-world design simulation increases the meaningfulness and credibility of the model.

4) The problem of action.

A popular COD convention is to treat actors as "agents" that respond to certain (predefined) communications with certain (predefined) routines. A signal to an agent triggers a specific response (or set of responses; or triggers it with a certain probability).

In contrast to this sanguine assumption, RWD takes the "agency problem" as a central issue. The tacit assumption underlying the design of most organizational structures is that clear communications, coupled with clear and certain rewards and punishments, are sufficient to ensure appropriate behaviors. To assume otherwise is to become quite pessimistic about the possibility of coordinating the behavior of large numbers of people without tremendous cost (e.g., in face-to-face meetings). Nonetheless, everyday occurrences continually challenge this assumption. People issue orders and requests that generate unexpected responses because the person receiving the signal interprets it differently than the sender intended. Such communications are more or less deeply context-dependent; thus real-world designers try to incorporate their native understandings of signalling contexts into their designs.

For real-world designers, the problem is: How can they be confident that a given signal will result in certain actions? Real-world designers have a rich set of schemas of agent

[7] ibid.

responsiveness and signalling contexts. These schemas help them analyze typical contexts and decide on which types of signals are better or worse triggers of which types of behavior. These schemas address: first, the motivations of particular receivers; second, the resources of the receiver; and third, the nature of the signal. The schemas help real-world designers approach the agency problem with a rich repertoire of solutions.

The receiver's motivation. One aspect of the signalling context is a presumption of the receiver's willingness to conform to the sender's intent. Some actors are known for their "initiative" and thus can be expected to take action given only *information*, rather than specific directions. At the other end of the initiative spectrum are agents who have a propensity to shirk from certain tasks. For such agents, simply conveying information is not enough; for those who are likely to not act, *formal orders* must be provided in addition to information.

Anticipating the interpretation of a particular signal-to-act thus involves inferring the likely state of the receiving agent. There are no general rules for addressing this issue; real-world designers' background knowledge about the organization provides the basis for such inferences. For instance, according to a member of the Electra organization design team, "Drafting won't do anything without a Work Order." Such a statement, necessary to specify the proper signal, is specific to Electra. There is no general statement about the relationship between Design Engineering demands, Drafting responses and work orders outside Electra. (Although slightly more general hypotheses may be available in the literature: for instance, that Design Engineers are often not receptive to signals from Manufacturing about the "manufacturability" of their designs.)

The receiver's resources, skills, etc. A second dimension is the receiver's capacity for responding to the trigger. For those likely to act, but in unpredictable ways, signals must include *directions* for action. The Electra design team distinguished between "process" problems, which stemmed from lack of triggers, and "training" problems, caused by a lack of understanding of how to respond to a trigger. A Draftsperson, for instance, may be able to respond adequately to a request for a revised set of paper drawings, but may not be able to fulfill a request for the same set of drawings to be set over electronic mail.

The nature of the signal. In addition to the propensity and ability of the receiver to conform to the signal, another aspect of the signaling context is the nature of the signal. One way Electra designers conceived of the force of the signal was in terms of the "hardness" of the information conveyed. "Hard information" was unambiguous and final — and which, for that reason, was presumably an effective stimulus of a particular behavior. Soft information, on the other hand, was known by both parties to be ambiguous and subject to change and thus more likely to allow for multiple interpretations and multiple responses.

The problem is that hardness is contextual. At another organization, for instance, design engineers gave preliminary cost estimates to sales personnel so the latter could plan their sales strategy. Once they had the information, however, some sales managers began using it to quote prices for customers, thus locking the

company into those estimates. What seemed like "soft," "guesswork" cost information to the engineer was interpreted as a formally binding cost estimate by a sales manager.

The problem is that one cannot always foresee the context in which information will be interpreted. Thus real-world designers sometimes create an artificial context by formally labeling information. Electra organization designers used formal procedures to bind a particular predefined interpretation to a particular predefined context. For instance, cost information was formally and explicitly labeled "hard" and "soft" according to the particular task which generated it. Further, real-world designers may attempt to manipulate the hardness of the information is specific contexts. "I'd like to firm up that info," said Jim, a member of the Electra team, as he proposed a rule that required a particular report to be interpreted as a final cost estimate, rather than "soft, waffley stuff." Such rules not only tells the receiver how to interpret the cost data. They also encourages the sender to be more careful about how their signal might be received.

Often, however, real-world designers cannot completely and confidently specify the nature of a signal. They thus resort to markers for such ambiguity. For instance, "Design tells Drafting to make changes to drawings," reads the Electra design; the designers could give no more specific advice about how to tell Drafting to do so.

Signals are invariably context-specific. In the COD paradigm, receivers' motivations and capacities are modeled and certain specific signals are assumed to have certain efficacy. Thus the COD paradigm assumes a particular context. For real-world designers, such issues are subject to continual revision, based on organization-specific knowledge. Real-world designers recognize that contexts shift. They respond to such uncertainty by binding specifiable contexts to particular signals; but such a strategy can only be partially effective.

Conclusions

Real-world designers do not necessarily use better strategies for solving problems of organizational representation. Neither are all of their strategies available for researchers working in COD. But by describing, typifying and understanding real-world design in natural settings, we might find some useful extensions to COD. The basic contrast between RWD and COD approaches, as suggested by Table 1 below, is that RWD trades rigor for richness and meaningfulness.

Both COD and RWD provide screens for deciding what features to model and which to ignore. COD often proceeds by ignoring features that cannot be modeled with available modeling techniques. RWD often proceeds by ignoring features that raise political problems, that violate organizational norms, or that raise issues that are not solvable by designers.

Both COD and RWD must choose a level of analysis, a level of granularity. For COD, this choice is typically embodied in an ontology that is then consistently implemented throughout the model. For RWD, granularity is a function of the natural-language ontologies of organizational members. It thus is more inherently meaningful, but also shifts from area to area.

Both COD and RWD have provisions for simulating and testing their models. COD

simulation techniques are quite rigorous compared to RWD techniques, which often rely on the vagaries of narrative simulation. Nonetheless, narrative simulation is a better guarantor that the model "makes sense" in a specific organizational context.

Both COD and RWD must address the agency problem, or the question of how the agents they model will respond to the signals they are provided. The signal-response set is dependent upon the context in which the signal is sent and received. COD designers assume a particular type of signal context. RWD designers, by contrast, have rich schemas of signal contexts they employ to analyze various contexts in detail; they attempt to use these schemas to bind certain contextual features to the signal, with more or less success.

As are most formal techniques, COD is susceptible to the criticism that gains to rigor come at the cost of a lost of meaningfulness. The contrast with RWD approaches amplifies such a criticism. It also, however, highlights some of the costs to meaningfulness: inconsistency, bias, and reification of organizational culture. Understanding the sources of these costs may help us to make COD more meaningful without suffering the same pitfalls.

Table 1: Comparisons between COD and RWD approaches to four problems of organizational modeling

Problem	Question	COD approach	RWD approach
Simplification	What to include?	Excludes what cannot be modeled by existing technique.	Excludes issues that raise political problems.
Granularity	What's the level/unit of analysis?	Consistent throughout the model.	Uses natural taxonomies of activities.
Simulation	How can we check the accuracy and consistency of our model?	Context-independent, rigorous testing of many aspects simultaneous.	Context-sensitive testing of features amenable to narration.
Agency	How can be certain that the agents in our model behavior as we assume they will?	Assumes a particular signalling context.	Uses rich knowledge of aspects of signalling context to bind particular features to particular signals.