USING AGGREGATION FOR AUTOMATIC IDENTIFICATION OF WORK PROCESSES IN THE MANAGERIAL HIERARCHY

Bahadır Kaan Özütam
Boğaziçi University, Computer Engineering Department
İstanbul, Turkey
E-mail: ozutam@yahoo.com

KEYWORDS

Aggregation, Modeling Organizations.

ABSTRACT

Aggregation is a qualitative reasoning technique which replaces repetitive cycles of process instances with a higher level description of a single continuous process. We investigate how this AI method can be used in the modeling and simulation of social organizations. Different levels of an organization such as first level employees, middle management, and top management view the available information using different degrees of abstraction. These different levels thus have different ontologies Our approach involves starting with descriptions of low-level work processes and using the automatic abstraction mechanism to come up with higher level process descriptions employing ontologies with fewer details. The output of our aggregator provides a suggestion for how a hierarchy of tasks might be constructed in that organization.

1. INTRODUCTION

The choice of the abstraction level is a critical decision in modeling any system. A low-level, high-resolution model is more realistic than a more abstract one, but dealing with such models may be both computationally expensive, and, more importantly, cognitively inappropriate. Some abstraction is inevitable. A social organization such as a large company is a multilevel structure in which each upper level is an abstraction of the lower ones. Different levels view the available information using different "ontologies." Finding the most suitable design for such hierarchies and determining the appropriate ontology for each level is an important problem in organization theory (Daft 2001).

Computer modeling and simulation are used with increasing frequency in the study of organizations (Davis et al. 2007). Agent-based and equation based models, system dynamics, cellular automata and social network models are some of the dominant trends. Among the early significant contributions to this field, one can list the Garbage Can Model (Cohen et al. 1972) of Cohen, March and Olsen, the evolutionary theory of economic change (Nelson and Winter 1982) by Nelson and Winter, and March's work (March 1991) on exploration and exploitation in organizational learning. More recent studies include the model of organizational demography and culture (Carroll and Harrison 1998) by Carroll and Harrison, Gavetti and Levinthal's work on cognitive and experiential search (Gavetti and Levinthal 2000), and the modeling of organization structure in unpredictable environments (Davis et al. 2006) by Davis, Eisenhardt and Bingham.

We apply the qualitative reasoning technique of aggregation of processes introduced by Weld (Weld 1986) to organizational modeling. The basic form of aggregation as explained in Weld's paper detects repeating cycles of discrete processes using the simulation history structure, and replaces them with a higher level of description, a continuous process which performs the same action as the processes in the repeating cycle. The process descriptions have two components; preconditions and changes. The preconditions define the conditions to be satisfied so that the process can be active. The changes are the effects that the process will create over other objects in the simulation when it is active.

In this ongoing work, we implemented and used the basic aggregation algorithm for automatic identification of higher-level work processes in an organization setup, improved the method so that it can be used more conveniently and effectively in this domain, and identified new aggregation types that seem to be required for organizational modeling.

2. MOTIVATION

Identifying the abstraction degrees of the information viewed by the different managerial levels, and designing the managerial hierarchy which is appropriate for a given organization are important problems in the study of organizations. The aggregation technique seems to be a promising method for these purposes. Having obtained the different level models, it is possible to simulate all of them and see how the predictions of the more abstract ones diverge from those of the lowest level. In this regard, we are interested in the following questions: Are the upper level ontologies suggested by the program indeed the ones seen in real-life multi-level organizations? Do the simulation results of the different level models differ, and if they do, how do they differ? Does this mean that one or more of the simulations are wrong? In which way do discrete and continuous simulations differ? Can we see the difference between discrete and continuous simulations when we run a simulation with aggregated, disaggregated and partially aggregated processes? We hope to gain new insight about identification of different level processes in organizations, and, indeed, about multiple-level modeling in any domain, from the answers of such questions.

Proceedings 21st European Conference on Modelling and Simulation
Ivan Zelinka, Zuzana Oplatková, Alessandra Orsoni ©ECMS 2007
ISBN 978-0-9553018-2-7 / ISBN 978-0-9553018-3-4 (CD)

3. WHAT IS AGGREGATION?

Discrete processes make atomic effects over the objects they affect, and the amount of this change is known in advance. If necessary, one can repeat the discrete process multiple times until the desired total amount of change is obtained. This effect can also be realized by a single instance of a continuous process, as will be demonstrated below.

Let us consider an example from our organizational model to illustrate what aggregation does:

Example 1: Consider three processes to produce a Product using some Raw Material:

SeizeMachine:
Pre: the production machine is idle
Ch: reserves the machine.
ProduceHalfProduct:
Pre: the machine has been reserved ,
a sufficient amount of raw material exists.
Ch: converts a specific amount of raw material to a corresponding
amount of half-product.
ProduceProduct;
Pre: the machine is reserved,
a sufficient amount of half-product exists.
Ch: converts a specific amount of half-product to a finished product,and release the machine.

If we have some amount of raw material at the beginning, the sequence "SeizeMachine, ProduceHalfProduct, ProduceProduct" will repeat until almost all the raw material is transformed to products. The aggregation of a prefix of the simulation history generates a continuous process:

CP1
Pre: RawMaterial exists,
productOrder>0
The machine is idle
Ch: convert RawMaterial to Product (Assuming one unit of product is produced by one unit of Raw Material)

CP1 runs when production machine is idle and there exists raw materials. It converts all the raw material to products.

The process replacing the cycle must be a continuous process. If a discrete process is used to replace the cycle in an unsophisticated manner, the new process will just make a greater, but fixed amount of change every time it is activated. The duration and the total effect of a continuous process, on the other hand, are not determined before it runs. The effects it creates are defined in terms of change rates per unit time and the total effects of each different instance of the aggregated continuous process are different from each other in general, since they depend on the model environment.

The processes that can be abstracted by aggregation need not all be discrete. Continuous processes can also be replaced. This allows the aggregator to find nested cycles, and make multiple replacements over the same set of processes.

Let us add a new process to our model: Deliver(*n*) runs when *n* products exist, and delivers *n* products to the customer. In this new setup, CP1 (aggregated from SeizeMachine, ProduceHalfProduct, and ProduceProduct, as before,) does not run alone. An instance of Deliver(*n*) runs each time a new batch of *n* products have been produced. The sequence "CP1, Deliver" is repeated until all the raw material is converted to products and delivered. In this case, the aggregator finds a new cycle "CP1, Deliver" and replaces it with a new continuous process CP2, which converts the raw materials to products and delivers them. CP2 represents the operations of a Production Department in this organization.

4. USING AGGREGATION IN ORGANIZATIONAL MODELING

Different levels of an organization such as first level employees, middle management, and top management are actually different abstraction levels, and the information requirements of each level differ from the others, depending on their interests. So the focus of the modeler must be the different levels of an organization and the ways these levels view things. The different levels have different ontologies. In the higher levels, some groups of processes of the lower levels are represented with more abstract processes, some low level decisions are not seen, but only the cumulative results of these decisions are visible, some variables of the lower levels are not represented, and individuals are viewed collectively as groups having aggregated characteristics. A consequence of the point of view that the higher level views are basically summaries of the more detailed lower levels is that we can obtain the whole model of an organization by repeated abstractions once we construct the lowest level of the model. Our aim is to realize this kind of modeling by an automatic aggregation process, which takes as its input a hand-made description of the lowest level.

The following working example illustrates the use of the aggregator described in the previous section to find such "natural" abstractions of the lowest level work processes. The higher level process descriptions output by the aggregator can be seen as a suggestion about how a hierarchy of tasks might be constructed in an organization with these low-level processes. *Example 2:* Consider a production company, whose lowest-level processes are defined as shown in Figure 1. The model is simulated with a scenario in which a customer is found and a production order of a certain amount is received from that customer. Having obtained the simulation history, the aggregator gave the replacements in the model shown in Figure 2.

An examination of the processes resulting after the second-level aggregation leads one to say that this organization can have five departments; CP1 = Purchasing, CP6 = Marketing, CP7 = Production, CP4 = Accounting, CP5 = Research and Development. Note

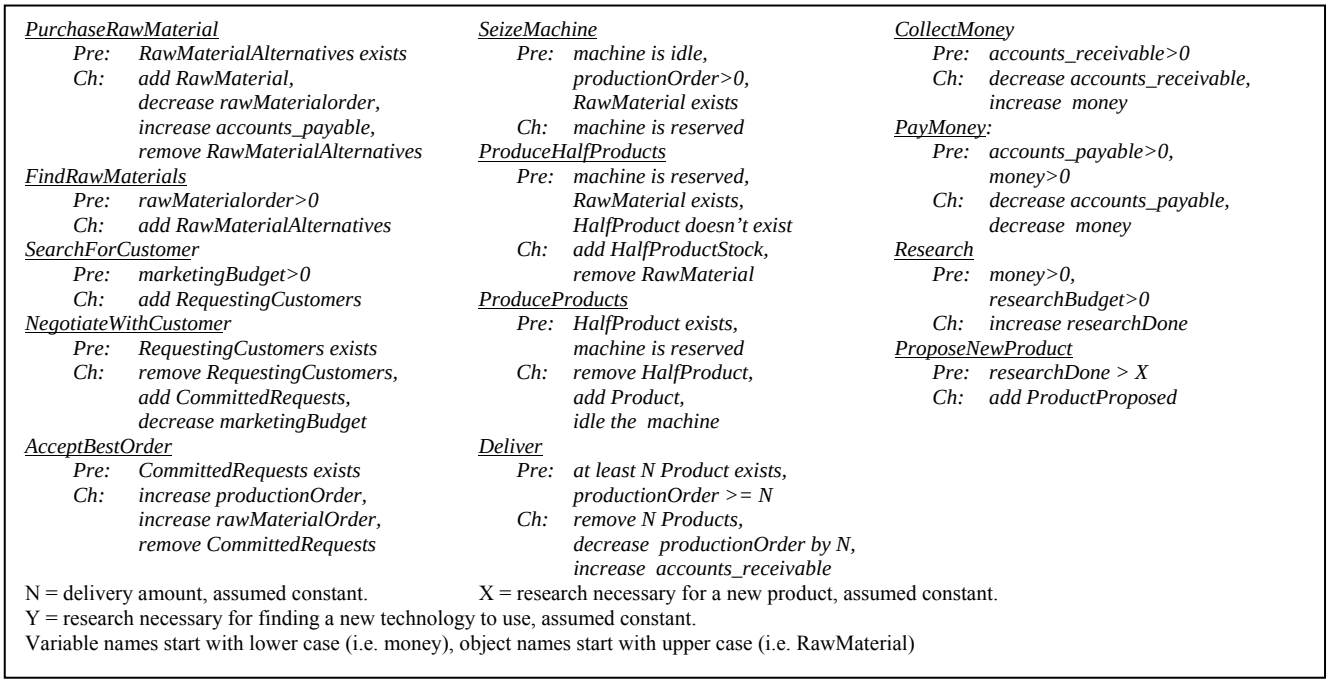

Figure 1 : The Processes in the Organization Model

that the aggregator output is not just a suggestion of a possible organizational structure. The descriptions of the high-level processes also indicate which ontologies, i.e.subsets of the set of objects and variables used in the lowest level model, are used by the respective "managers" of the departments when they talk to their own superiors. For instance, the object type "Half-Product" is not mentioned when the Production Department's manager communicates with top management.

5. NEED FOR FURTHER AGGREGATION METHODS

We took Weld's algorithm (Weld 1986) as a starting point in constructing the aggregator whose runs were exemplified in Sections 3 and 4. Even in the simple case of Example 2, Weld's algorithm has problems in identifying cycles, since multiple "departments" are supposed to run in parallel. The improvements we incorporated to the algorithm to handle such cases are explained in Section 5.1.

The full realization of the goal of automatic identification of work processes in the managerial hierarchy requires more than even the improved version of the cycle aggregation method we considered until now. Consider Example 2. The highest aggregated level, which is the furthest one can go and obtain sensible results using this method, includes five continuous processes. This is not the most abstract level one would hope to obtain. Imagine the owner of the company, who is not concerned with the internal operational details of the company, but is only interested in the performance measures of the company, rather than how they are achieved. There must be a highest level process which defines the entire organization in terms of the changes it caused, and maybe more levels above the output of Example 2 under this top level. We therefore need further methods of aggregation to complete the construction of the managerial hierarchy of the organization. One such method that we examined is explained in Section 5.2, including a working example. Other, as yet unimplemented, ideas that we plan to look at, are explained in Section 6.

5.1. Improvements in the algorithm

In many cases, Weld's algorithm may detect multiple alternative cycles as aggregation candidates in the simulation history. To be able to prefer better alternatives, the aggregator can be improved to choose the cycle according to certain principles. A company has several departments. Usually, mutually irrelevant tasks take place in different departments. Descriptions of such mutually irrelevant process instances which run in the same time will be printed out close to each other in the output of a sequential simulation (like the ones carried out by our process simulator) of the entire company, such that they may appear as candidates for being aggregated together to a naïve algorithm.

Consider the sequence CollectMoney, PayMoney, CollectMoney, PayMoney, … and a separate sequence FindRawMaterials, PurchaseRawMaterial, FindRawMaterials, PurchaseRawMaterial, … Although the two departments perform mutually irrelevant tasks, they operate simultaneously. When the aggregator analyses the history, it may find several repetitions of the sequence "CollectMoney, FindRawMaterials,

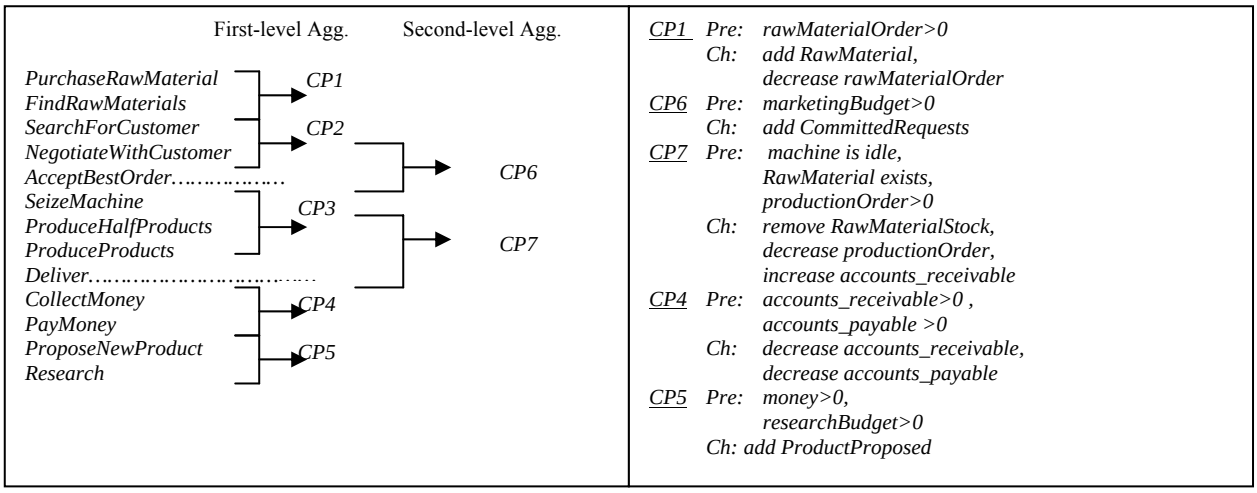

Figure 2 : First and Second Level Aggregation of the processes in the organization model

PayMoney, PurchaseRawMaterial", and try to replace this "cycle" of four with a single continuous process, and this is clearly not what we want it to do. (This predicament of distinguishing irrelevant processes is referred to as the local evolution problem (Forbus 1993)). How can the aggregator distinguish these mutually irrelevant processes? Let us define what we mean by "mutually irrelevant" in terms of the aggregator's input.

Definition : A process A is a *change predecessor* of a process B if at least one of the changes of A affects a variable referenced in the preconditions of B.

Definition : Two processes are *mutually irrelevant* if neither of them is a change predecessor of the other one.

The main idea is that the processes in a cycle must trigger each other so that they form a meaningful cycle. Irrelevant processes may occur successively in time by chance, in which case they do not form a meaningful cycle.

What if a nonzero duration of time passes before the start of B after its change predecessor A terminates? Can we still say that A triggers B, and that they are good candidates for a cycle? If A triggers B, changing a variable which is referenced in the preconditions of B, B must be activated just when A changes that variable. If there is a gap between these two events, there must be another event occurring in this gap which triggers B. Thus, if this event had not occurred, B would not be active. A does not trigger B, and they are not good candidates for a cycle.

In our simulator, time proceeds in ticks, and the amount of time between two successive time ticks is the unit time (smallest amount of time that must be considered) for the simulation. The starting and ending times of processes correspond to starting and ending of time units since there can be no smaller time in the simulation (A discrete process lasts one tick). Thus, if a process triggers another process, the triggered process must start just at the tick at which it is triggered.

Definition : An instance of a process A is a *time predecessor* of an instance of another process B, if the instance of B starts just when the instance of A terminates.

If the aggregator only accepts the instances of processes which trigger each other, it will avoid including irrelevant processes appearing in one cycle and will form meaningful cycles.

Definition : An instance A_i of a process A *triggers* an instance B_i of another process B, if A is a change predecessor of B and A_i is a time predecessor of B_i.

Must the last process of a cycle trigger the first process of the cycle? Considering the meaning of the word "cycle," one is tempted to say "yes". But consider the case where we produce products from raw materials. The process will continue as long as we have raw materials and a nonzero production order, despite the fact that the act of finishing a product does not trigger the start of another production. Thus, we have decided to accept sequences in which every process except the first is triggered by the previous one as valid cycle iterations, even when the last process of such a sequence does not trigger the first process. Example 2 was run on our implementation of an improved aggregator in which these augmented criteria for cycle detection enabled the identification of the sensible hierarchy seen in Figure 2.

5.2. Superclass Aggregation

The aggregator described above recognizes a cycle only if multiple instances of the same process are seen to be repeating. In some cases, abstraction of two non-identical but sufficiently similar process descriptions to a "superclass" process may help. Consider the sequence given below for a cleaner who is working in a building:

```
ProduceNewProduct
      Pre:  RawMaterial exists,
            newProductOrder > 0
      Ch:   convert RawMaterial to  NewProduct
DeliverNewProduct
      Pre:  at least N NewProduct exists,
            newProductOrder >= N
      Ch:   remove N NewProducts,
            decrease  newProductionOrder by N,
            increase  accounts_receivable
N = delivery amount
```

Figure 3 : The processes added in Example 3

Cleaning the floor,
Walking down the stairs one floor,
Cleaning the floor,
Going down one floor by elevator,
Cleaning the floor, …

No proper repetition can be found. But if we abstract "Walking down the stairs one floor" and "Going down one floor by elevator", to the new process "Going down one floor", we can obtain the new sequence

Cleaning the floor,
Going down one floor,
Cleaning the floor,
Going down one floor,
Cleaning the floor, …
in which a repetition can be detected easily.

Forbus and Falkenhainer give a good example of how to compare processes to find out their similarity in their work on analogical processing with the Structure-Mapping Engine (Falkenhainer et al. 1990). The idea is to accept two processes as subclasses of a superclass if a significant proportion, rather than all, of their properties are identical. This operation can of course create problems if the non-identical features of the low-level processes, which are abstracted away, play important and different roles in the actual system. This method of abstraction can therefore sometimes produce incorrect higher-level models. The allowed error can in fact be tuned by the user, since the aggregation algorithm decides whether to create a superclass process for two given low-level processes according to a similarity function which measures the match between two processes. Processes whose similarity degree exceeds a user-defined constant are aggregated. Keeping the required similarity degree high will avoid the kind of error explained above, with the cost of a narrow scope for aggregation. Keeping it low will allow more processes to be aggregated together, and the divergence of the higher-level model predictions from the lowest level ones will increase. Upon receiving the input set of process definitions before beginning the simulation, the aggregator first tries to identify superclass processes among these. Simulation with the aim of cycle detection is then performed with the updated process list. Since common superclasses can be detected even among the newly abstracted processes during the higher-level aggregations, the aggregator runs this similarity detection procedure again whenever new process descriptions are added to the list.

Example 3: Consider adding two new processes which are similar to some already present processes to the model of Example 2: The company has obtained a new machine of type "NewMachine", which uses the same type and amount of raw materials, but produces a different product of type "NewProduct" directly, without going through the half-product stage. The new processes are shown in Figure 3.

We submitted this model to the aggregator with a scenario of receiving orders for Product and NewProduct randomly with the same probability. Two different runs, with required similarity degrees of 0.9 and 0.6, respectively, were performed.

In the first run with a required similarity degree of 0.9, shown in Figure 4, the cycles the aggregator replaced in the first-level aggregation were the same as in Example 2, and ProduceNewProduct and DeliverNewProduct were also aggregated to a new process. The second-level aggregation had the same results as in Example 2, and the new production process was not aggregated any further.

In the second run, with a required similarity degree of 0.6, the aggregator first found out that Deliver and DeliverNewProduct are similar processes. Similarly, after generating CP3 from "SeizeMachine, ProduceHalfProduct, ProduceProduct", it also recognized CP3 and ProduceNewProduct as similar processes. Consequently, the program made the aggregation shown in Figure 5, with the process definitions given in Figure 1 and the production "department" process of the organization came out to be isomorphic to the one in the Example 2, (Figure 2)

As seen in Figure 5, the Deliver and DeliverNewProduct processes were replaced by the more general process CP10 and ProduceNewProduct and the new process CP3, which was produced by the aggregation of SeizeMachine, ProduceHalfProduct and ProduceProduct were replaced by the more general process CP11.
The aggregator found the similarity of Deliver and DeliverNewProduct to be 0.67, and the similarity of CP3 and ProduceNewProduct to be 0.72, causing both pairs not to be abstracted together in the first run, and to be abstracted together in the second run. Both results may be preferable depending on the requirements of the context. When analysis considering the sales of the products individually is required, seeing the products as separate units and getting the results accordingly will help. In another case, for example, when the total sales of the company, or only the total amount of money made by the company is required, seeing the products as equivalent and looking only at the totals will help.

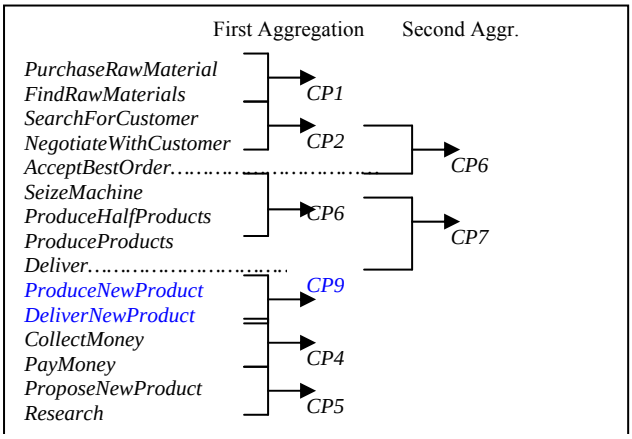

Figure 4 : Aggregation in Example 3 when the required similarity degree is 0.9

6. FUTURE WORK

More tools are necessary for a complete and sensible identification of and reasoning about the managerial hierarchy of an organization. In this ongoing study, we plan to implement the following additional methods.

6.1. Parallel Aggregation

Weld (Weld 1986) defines parallel aggregation as "the replacement of multiple instances of the same process occurring simultaneously", without including the working principles or any example. This idea promises to serve as wide a range of possibilities as serial aggregation does, so we plan to realize it. While serial aggregation makes a vertical replacement in the history of the simulation since it replaces instances spread over time, parallel aggregation makes a horizontal replacement, since it replaces process instances occurring in the same time, but spreading over the actors of the simulation.

In parallel aggregation, as well as the starting time, we know the ending time since only one iteration occurs. But we do not know the total effects, since we do not know how many parallel processes will participate in this parallel processing in different moments of the simulation. The definition of parallel aggregation requires defining a new type of process, since neither the discrete nor the continuous process definition formats we use meet the requirements of the process generated as a result of a parallel aggregation. This new type of process occurs atomically, and its termination time is determined when it starts. But unlike discrete processes, its total effects can not be known before simulation, since it depends on the number of parallel process instances participating which can be different in different moments of the simulation. As a result of this, the total effects of the process change in different instances.

6.2. Reverse Aggregation

Consider the case where we are given the abstract description of an organization, or some components of

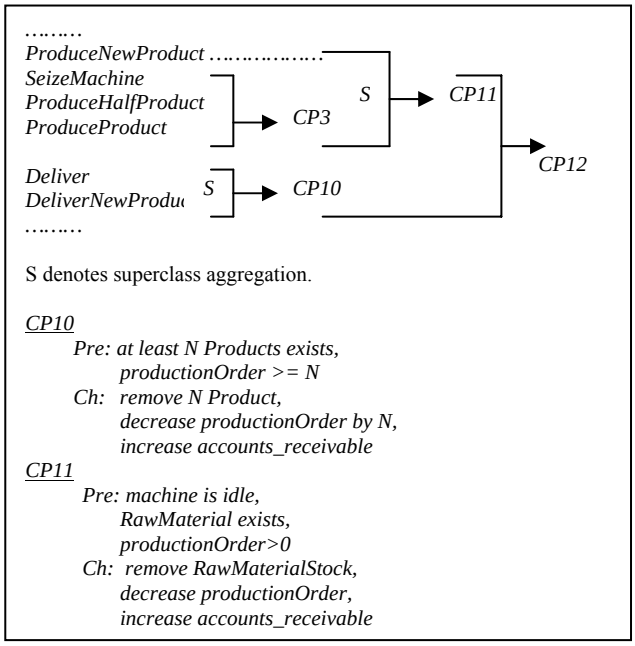

Figure 5 : Aggregation in Example 3 when the required similarity degree is 0.6

the organization, and we attempt to obtain the descriptions of the lower level processes. For instance, we may try to guess the invisible structure of a rival company from the partial information that is available about that company. What we need to do is to "reverse aggregate" the high-level models.

At first sight, reverse aggregation seems to be a hopeless task, since it involves creating a more detailed description than its input. But if we also have a library of common lowest-level process descriptions, we might at least guess a candidate input process list which would, when aggregated, result in the higher-level models that have been given to us.

6.3. Mixed Levels of Abstraction

Abstraction is meaningful if we do not need the detailed information we lost, or we have the ability to reconstruct it in case we need it later. Consider a manager who manages several working teams including several workers. The manager would not care about the workers individually, and would not want to know the details of each worker. In his daily operation, he would rather know the output of the teams, leaving the details of what happens within the teams to team leaders. The point of view of this manager is an aggregated level, which hides the first level employees inside teams. But one day, in case a problem about an individual first level employee occurs and affects the operation of the team, or even the whole organization, the manager would want to understand the case and need the details about the operation of the first level employee. This would require the manager to adopt, for the purposes of this case, a "mixed-level" model, which does not necessarily include all the complexity of the lower level processes which are

irrelevant in this case, but which contains the details necessary for the present reasoning task. We believe that the infrastructure we are preparing for the application of the methods described above can easily be adapted for examining such mixed-level modeling tasks.

6.4 Aggregation of Probabilistic Models

One of the simplistic aspects of Example 2 is that all processes are deterministic; e.g. once its preconditions are satisfied, ProduceProducts does its job, assuming that the machine will never fail. This, of course, is unrealistic. We plan to consider an alternative process format which supports probabilistic models like one where the machine in example 3 can be stipulated to have a specific fault probability, and the results of such eventualities can be described separately in the process descriptions. An aggregation algorithm for such probabilistic models may produce higher-level models which are themselves probabilistic.

7. CONCLUSION

The hierarchical structure of an organization such as a large company provides a suitable domain for investigating aggregation, since the hierarchical levels composed of various positions (first level employee, middle management, CEO, etc.) in the organization correspond to different aggregation levels. Our experiments so far with our improved version of Weld's aggregator have helped us identify some other tools that are necessary for reasoning about these issues in this domain. We are actively working on the design and development of these tools.

REFERENCES

Carroll, G., and Harrison, J. R. 1998. "Organizational Demography and Culture: Insights from a Formal Model and Simulation" *Administrative Science Quarterly*, 43: 637-667.

Cohen, M. D., March, J., and Olsen, J. P. A. 1972. "Garbage Can Model of Organizational Choice" *Administrative Science Quarterly*, 17(1): 1-25.

Daft Richard L. 2001. *Essentials of Organization Theory and Design.* Ohio: South-Western College.

Davis, J., Eisenhardt, K. and Bingham, C. 2006. "Complexity Theory, Market Dynamism, and the Strategy of Simple Rules". Stanford Technology Ventures Program -- Working Paper.

Davis, J., Eisenhardt, K. and Bingham, C. 2007. "Developing Theory Through Simulation Methods". *Academy of Management Review,* 32(2): 480-499

Falkenhainer, B., Forbus, Kenneth D. and Gentner, D. 1990. "The structure-mapping engine: algorithm and examples", *Artificial Intelligence* 41: 1-63.

Forbus, K. D. 1993. "Qualitative Process Theory", *Artificial Intelligence*, 59: 115-123.

Gavetti, G., and Levinthal, D. 2000. "Looking Forward and Looking Backward: Cognitive and Experiential Search." *Administrative Science Quarterly*, 45: 113-137.

March, J. G. 1991. "Exploration and Exploitation in Organizational Learning" *Organization Science*, 2(1): 71-87.

Nelson, R. R. and Winter, S. G. 1982. "An Evolutionary Theory of Economic Change." *Cambridge, Massachusetts: Belknap - Harvard University Press*.

Prietula, Michael J., Kathleen M. C. and Gasser, L. (ed) 1998., "Simulating Organizations: Computational Models of Institutions and Groups", *AAAI Press /MIT Press*.

Weld, Daniel S. 1986. "The Use of Aggregation in Causal Simulation", *Artificial Intelligence*, Vol 30: 1-17.