A FORMAL ENTERPRISE MODEL FOR PROJECT-ORIENTED MANAGEMENT OF BATCH PLANTS

M. M. CANAVESIO
UTN - Facultad Regional Santa Fe
Lavaisse 610 – Santa Fe (S3004EWB) – Santa Fe - Argentina
e-mail: mcanaves@frsf.utn.edu.ar
E. C. MARTINEZ
CONICET - Instituto de Desarrollo y Diseño
Avellaneda 3657 – Santa Fe (S3002GJC) – Santa Fe – Argentina
e-mail:ecmarti@ceride.gov.ar Fax: + 54 342 4553439

Abstract

Intense competition and rapid environmental changes are revealing severe limitations and drawbacks of the hierarchical and functional-oriented management system currently used by the vast majority of batch plants. In this work, a formal modeling framework for batch plants using projects and situation calculus is proposed. Projects are autonomous, temporary entities within the enterprise, in which different types of expertise are combined to achieve a concrete set of goals or deliverables. Situation calculus is used to represent the plethora of dynamic relationships involving projects, resources, actors etc. at different levels of abstraction over time. The use of a formal language permits to define the enterprise model specifications for rigorous reasoning about batch plant structure and behavior. The proposed framework is exemplified with a case study comprising a dairy facility that involves the production of 100+ different fresh products using an order-driven management system.

Keywords

Batch Plants, Enterprise Modeling, Make-to-order, Situation Calculus.

Introduction

Intense competition in the batch process industries due to economic globalization and market fragmentation gives rise to shortening of each product lifecycle and rapid changes in the enterprise environment. This situation is forcing batch plant managers to develop the ability to quickly produce customized products. Batch plants manufacture a variety of products that should satisfy stringent end-use properties. End-use properties are product attributes that characterize its intended use and are a direct measure of its value. Reproducing end-use properties is also a key requirement. For example, crystal shapes in a pharmaceutical product determine the bio-availability of the active substance. The latter is the end-use property that values the product, whereas obtaining the right shape is a process requirement rather than a product attribute. Increased customization to satisfy end-use properties for each customer causes a constant enlargement of product portfolio due the incorporation of entirely new recipes or the creation of slightly different versions of existing recipes.

Current organizational designs of batch plants are based on a rigid and "Tayloristic" organizational model. This is a big obstacle in the way to achieve higher degree of flexibility and reactivity to cope with the increasing rate of change and complexity surrounding most manufacturing enterprises. To ensure reactivity and flexibility, new forms of decentralized management in which separated functions (planning, scheduling, maintenance, product development, etc.) are closely coordinated around a common objective need to be created (Schweyer and Haurat, 1997). Different

organizational units should work together in a team-like, goal-oriented environment. Enterprises are turning towards new decentralized yet cohesive forms of organization like project management approaches (Martinez, 2000). In a multi-project management approach, each project can be seen as an autonomous, temporary entity within the enterprise, in which different types of expertise are combined to achieve a concrete goal or deliverable (e.g. completing a given order, product development, maintenance program, etc.). More specifically, for the purpose of production planning and scheduling each customer's order is managed as an individual project (Martinez and Perez, 1998), which can be further decomposed into sub-projects.

Enterprise Model

Conceptual Design

An enterprise model is a computational representation of the structure, activities, processes, information, resources, people, behaviors, goals and constraints of an enterprise (Fox and Gruninger, 1997). The enterprise model that is presented here is based on project-like organizational units. The project-oriented approach induces a recursive management structure for batch plants. This approach presents two advantages: (a) Any given project is a unit having self-regulatory and self-organizing characteristics that in turn can contain further sub-systems and so on. (b) It lends itself to decentralized decision making. The project-oriented structure thus defined is both a generator of efficiency and absorber of complexity to embed a highly adaptive and responsive management structure.
In this model, each project is an autonomous, temporary entity within the plant, in which new instances of roles, actors and relationships are created for achieving project goals and at the same time the plant goals. Anywhere in the abstraction hierarchy, the enterprise model defines the following basic concepts:

- In each project different instances of actors and roles are defined. An *actor* is anything that is capable of executing certain activities in a batch plant. Actors are human beings or artificial agents. A *role* involves a set of responsibilities and actions carried out by an actor or a group of actors within the plant. Actors play roles. Different actors can play different roles at different moments of time. One role instance is a project-manager role defining the responsibility for project execution and the other instances are the project-team that performs the work. Actor communication is performed by sending messages among them. Actor *relationships* are: `client-server`, where roles such as project-manager, project-leader are *clients* and roles such as resource-manager are called *servers*, which are in charge of processing request for resources usage and availability of functions, `reports-to` (among project leaders or clients), `responsible-for` (among client actor and project), `provided-by` (among requests and server actors).
- Each project is goal-oriented, so it must produce a specific deliverable for a clearly identified client and might need as inputs the deliverables of one or several other projects. A project goal is a desired state of affairs. Project goals become responsibilities of roles and the actors playing these roles. Project goals are management goals or operational goals.
- A set of temporally available *resources* or *skill centers* (including equipment items, operators and utilities) is provided. A *resource* or *skill center* is an homogeneous set of equipment items, human operators, skills, and utilities. These skill centers will either work on different projects or develop tools and methods for their domains.
- A library of objects containing project-relevant *knowledge* and *information* (master recipes, historical records from previous similar projects, maintenance data);
- A life cycle covering six ever-present phases in carrying out an activity: (1) Specification-What? (2) Planning-How? (3) Assignment- Who? When? (4) Execution (5) Efficacy control-Did we make the right thing? (6) Efficiency control – Did we work correctly? Each one of these phases can be seen as another independent project. Each particular instance of the project is in itself a partially ordered set of activities conducive to a sub-objective that can be in turn an input to another project process instance. All the links between project at the same level and the hierarchical links between project with its super-project (parent) or with its sub-projects (children) are called *relationships*.

The enterprise model concepts are defined formally by writing axioms in an extension of situation calculus. These axioms capture *semantic* of each concepts.

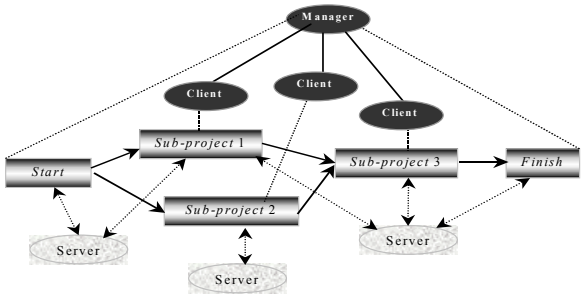

Figure 1. Conceptual Design of Enterprise Model.

Situation Calculus

The situation calculus (Reiter, 2001) is a second-order language ϑ for representing dynamically changing "worlds". All changes to the world are the result of named *actions*. A sequence of actions that represents a possible world history is called a *situation*. The constant S_0 is used to denote the *initial situation*. The term *Do(a,s)* denotes the situation that results from the execution of action *a* in situation *s*. In the situation calculus, actions are denoted by function symbols, and situations are also first-order terms. Relations whose truth-values may differ from one situation to another are called *relational fluent*. They are denoted by predicate symbols having a situation term as their last argument. Similarly, functions whose values vary from situation to situation are called *functional fluents*, and are denoted by functional symbols taking a situation term as their last argument.

An action is specified by first stating the conditions under which it can be performed by means of a *precondition axiom* of the following form: $Poss(a(x), s) \Leftrightarrow \Psi_a(x,s)$, where $\Psi_a(x,s)$ is a formula specifying the preconditions for action *a(x)*. Then, one must specify how the action *a* affects the state of the world with its *effect axiom*. It provides the causal laws for the domain of application. *Successor state axioms* are derived for each fluent.

The union of the following set of axioms will specify formal models:

- Axioms describing the initial situation, S_0.
- Action precondition axioms, one for each action.
- Successor state axioms, one for each fluent.
- Unique names axioms for the primitive actions.

These sets of axioms represent dynamic changes in the batch plant situations as a result all actions over time; this history of such changes begins at some initial situation of the plant.

In the next section, a formal enterprise model for a batch plant based on project is presented. Model concepts are defined using an extension of situation calculus.

Formal Model

A formal enterprise model defines rigorously and precisely (Koubaraskis and Plexousakis, 2000) concepts such as actor, role, project, resources and relationships among them. An advantage of formal models is that they are self-consistent and have certain properties. For instance, one can prove formally that all responsibilities assigned to a role are fulfilled, and resource constraints are maintained as a result of project execution. The complete set of axioms that defines the formal enterprise model allows managers to answer questions such as: What is it doing?, Who is it doing? When and Where is it doing? How and Why is it doing?. The formal model allows reasoning about logical specifications and plant constraints. The set of formal axioms has been implemented in the logic programming language Eclipse Prolog so as to simulate plant situation changes over time. For example, for an initial situation in which no order exists in the plant, the simulation output may reveal bottlenecks or dead-ends that call for changes in the operational logic. Also, the formal model simulation permits testing alternative designs for operational rules and shop-floor procedures.

All enterprise model concepts are defined formally by introducing appropriated constructs of language ϑ and writing axioms that capture their semantic. Thus, actions are denoted by function symbols, situations are first-order terms, and objects that represent everything in a batch plant are represented by logical predicates. The concepts of actor, role and their relationships are introduced by the following logical predicates *Actor(x), HumanActor(x), ArtificialActor(x), Role(y), PlayRole(x,y,z)*.

A project is formally defined by an expression of the following form:

Project *x*
  Purpose goals
  Attributes attributes
  Plan Preconditions $\Omega_p(x,S_0)$
  Activities Plan ϑ_1
  Role roles

EndProject; where *x* is a project identifier, *goals* and *attibutes* are axioms that describe goals and attributes of project. *Plan Precondition* is a set of axioms that defines what must be true in batch plan at the beginning of project activities. These preconditions are the result of actions that are performed by other active projects in the current plant situation. *Project Plan* is an ordered set of activities that will be executed for achieving project goals. *Roles* are axioms that assign actors to roles that are involved in the execution of the project plan. Roles are defined by the following expressions:

Role id
  Responsibilities resps
  Action actions

EndRole, where *id* is a role identifier, *resps* is a list of role responsibilities, a*ctions* is a set of axioms that describes actions which are performed by actors who play *id* role. Formally, an action is introduced by an expression of the following form:

Action a
  Preconditions $\Psi_a(x,s)$,
  Effects $\vartheta_a(x,s^*)$

EndAction,

where *a* is an action, and $\Psi_a(x,s)$ is a set of axioms that describes the fact that action *a* can be executed in situation *s* and $\vartheta_a(x,s^*)$ are axioms that specify how the current situation *s* is affected by performing an action *a*. Thus, a new situation *s** is achieved.

Case Study

The production plant (White, 1998) being studied is a dairy facility involving the production of 100+ different fresh products using a driver-order management system.

To complete an order, a number of batches are released to the shop floor for their processing. Each batch follows a sequence of five steps: preparation, quality testing, filling, packaging and delivery. Each order has a product type, batch size, a product container size, number of operations, different times per virtual line and one or more due dates. The sizes and capabilities for preparation units vary considerably to allow for greater concurrency. Once each batch has been formulated; a sample is taken and analyzed in the laboratory. If it is required, a correction is made, and a new sample is drawn for quality testing. This reformulation-test cycle continues until the batch passes the quality test. Once the batch is released it continues to sit in the formulation/mix unit until all of the necessary fill out equipment items (called "fill-out-train") are available. Often, a batch released must wait for a fill-out train, thus, holding up the use of the formulation/mix unit. This enforced delay simultaneously delays the batch, ties up the unit, and increases the work-in-process. Once all of the equipment in the train is assigned, the batch moves through it, then all of the pieces of equipment used are cleaned and made ready to use for another batch.

For this plant, examples of roles are: project-manager, project-leader, resource-manager, order-manager, etc. Each role has assigned responsibilities, for example, project-manager role should schedule the number of batches needed to complete an order; she is responsible for finishing the order at a point in time that is a close as possible to its due date, etc. Each role responsibility is performed by an action. These actions are formally denoted by function symbols. Some plant actions may be:`To_add_a_new_order`,`To_repair_the_schedule`,`To_break_down_production_order`,`To_make_quality_test`, etc. For each action should be defined a set of precondition axioms (Fig.3) and a set of effect axioms. Each new action that is performed at shop floor results in a new situation and this situation may change resource states. Resource states are denoted by fluent values. A resource state change is described by a set of successor axioms for each fluent (Fig.4). Situations may be *"Order#1 is released to the floor", "Batch sample #2.3 is queuing in the quality laboratory", "Released batch#2 is waiting to be filled out", "Batch#3 is waiting for an operator that connects the fill-out-train and initiates the fill-out operations"*, etc. All of these situations are the result of a sequence of actions performed by actors in the plant. Enterprise model specifications for the case study involve a set of axioms that formally defines a plant initial situation (Fig.2), projects with their actors, roles, actions and relationships that allow projects to achieve their goals.

Conclusions

A formal enterprise model for batch plats based on projects has been presented. This model can be used to specify a detailed design for information systems in batch plants. The model architecture is based on project-like organizational units. Each project is an independently acting self-similar unit within the enterprise. Formal enterprise modeling is based on an extension of situation calculus. The formal language allows representing the multiple-relationships among projects, resources, and actors at different levels of abstraction over time in formal way. The specifications of the enterprise model can then be formally verified for correctness.

PlayRole(Peter,Project_leader,Order#1);
PlayRole(resource_agent,resource_manager,Order#1);
Formul_Unit(u_1,Ready);Formul_Unit(u_2;Occupied)

Figure 2. Example of initial situation axioms.

$$\begin{aligned}&\text{POSS}(\text{TO\_OCCUPY\_UNIT\_FORM}(u, proj, p), s) \Leftrightarrow \\ &\quad \text{Ready}(u,s) \wedge \text{Clean}(u,s) \wedge \text{Empty}(u,s) \wedge \neg\exists \\ &\quad \text{maintenaice\_task}(u,s) \wedge \text{assigned\_task}(u, proj, p, t).\end{aligned}$$

Figure 3. Precondition axioms of an action.

$$\begin{aligned}&\text{OCCUPIED\_FORMULA\_UNIT}(u,id\_proj,p,do(a,s)) \Rightarrow \\ &\quad a = \text{to\_occupy\_unit}(u,id\_proj,p) \vee \text{occupied\_unit} \\ &\quad (u,id\_proj,p,s) \wedge a = \text{operation\_suspended}(u,id\_proj,p) \vee \\ &\quad \text{occupied\_unit}(u,id\_proj,p,s) \wedge \\ &\quad a = \text{to\_make\_quality\_test}(u,id\_proj) \vee \\ &\quad \text{occupied\_unit}(u,id\_proj,p,s) \wedge \\ &\quad a = \text{waiting\_for\_fill-out-train}(p).\end{aligned}$$

Figure 4. Successor State Axiom of a fluent.

References

Fox, M.,Gruringer,M., 1997, *Enterprise Modeling*. Department of Mechanical and Industrial Engineering University of Toronto. http://www.ie.utoronto.ca

Koubarakis,M, Plexousakis,D, 2000, A Formal Model for Business Process Modeling and Design, *Conference on Advanced Information Systems Engineering.*

Martinez, E.,Perez,G., 1998, A project-Oriented model of Batch Plants.Comp&Chemical Eng.V.22, N. **3**, pp.391-414.

Martinez, E.C., 1999, Project-Oriented Management of Batch Plants. Information System Design, *Comp&Chemical Eng Supplement*, pp. S711-S714.

Martinez, E.C., 2000, Systems thinking and functional modeling of batch process management using projects, *Comp&Chemical Eng*, V **24**, pp. 1657-1663.

Reiter, R., 2001, *Knowledge in action: logical foundations for specifying and implementing dynamical systems*, MIT Press, Cambridge, Massachusetts.

Schweyer, B. & Haurat, A.1997, Information system design using a project approach. *Journal of Intelligent Manufacturing*, N **8**, pp. 15-29.

White, C.H. 1989, Productivity and analysis of a large multiproduct batch processing facility. *Comp&Chemical Eng*, V 3, N **1-2**, pp. 239-245.