Goal-Driven Analysis of Process Model Validity

Pnina Soffer[1] and Yair Wand[1,2]

[1] Haifa University, Carmel Mountain, Haifa 31905, Israel
{wand,spnina}@mis.hevra.haifa.ac.il
[2] The University of British Columbia, Vancouver, Canada
yair.wand@commerce.ubc.ca

Abstract. Business process modeling and design, which has attracted much attention in recent years, emphasizes mainly graphical representation, usually without an underlying theory. The lack of a theoretical foundation causes several important issues to remain intuition- rather than theory -based. In particular, many process-modeling methods, being semi-formal, lack a mechanism for verifying the "correctness" of a process in terms of completeness, consistency, and feasibility. The paper proposes a generic theory-based process modeling (GPM) framework and criteria for validity evaluation of process models. The framework, which is based on Bunge's ontology, is formal and notation-independent. Validity is defined as the possibility of the process to achieve its goal. The paper discusses and characterizes causes for process invalidity and suggests ways to avoid these situations. The concepts of the framework and their usefulness for evaluating the validity of process models are demonstrated by applying them to a process taken from the Supply-Chain Operations Reference-model (SCOR).

1 Introduction

Business process modeling and design has attracted much attention in recent years in the context of integrated information systems and Business Process Reengineering.

Numerous methods for process modeling and representation have been proposed (e.g., Business Modeling Language (BML) [9], Event-driven Process Chains (EPC) [15]), addressing different aspects of processes, such as activity sequencing, resource allocation, and organizational responsibility for execution.

Process goals are included in many definitions of business processes (e.g., "a business process is a set of partially ordered activities aimed at reaching a goal" [7]). However, as opposed to process structure, the notion of goal has received relatively little attention in the literature. There are two main outcomes of this situation. First, goals are often viewed as external concepts, not integrated with process models. Second, most process modeling methods focus on graphical representation, usually without an underlying theory. The lack of a theoretical foundation causes several important issues to remain intuition- rather than theory -based. In particular, most process-modeling methods lack a mechanism for verifying the "correctness" of a process model in terms of completeness, consistency, and feasibility. An exception is some workflow-related process models, which use formalisms (e.g Petri-nets) and apply verification mechanisms [20, 21]. However, verification of workflow models addresses a limited set of aspects of the process, mainly activity sequencing.

This paper proposes a Generic Process Model (GPM), which is a theory-based, formal and notation-independent process modeling framework. The framework provides clear-cut criteria for evaluating model validity based on the integration of goals into process models.

We start by presenting the theoretical framework, then develop validity criteria and demonstrate their application to a process taken from the Supply Chain Operations Reference-model (SCOR) [16].

2 Generic Process Model Framework

In this section we present a theoretical framework for process modeling based on Bunge's ontology [3, 4], as adapted for information systems modeling (e.g., [14, 22]) and for modeling business process concepts [17, 18].

According to the ontological framework, the world is made of *things* that possess *properties*. Properties can be *intrinsic* (e.g. height) to things or *mutual* to several things (e.g. a person works for a company). Things can compose to form a *composite* thing that has *emergent* properties, namely, properties not possessed by the individuals composing it. Properties (intrinsic or mutual) are perceived by humans in terms of *attributes*, which can be represented as functions on time. The *state* of a thing is the set of values of all its attribute functions (also termed state variables). When properties of things change, these changes are manifested as state changes or *events*. State changes can happen either due to internal transformations in things (self action of a thing) or due to *interactions* among things. Not all states are possible, and not all state changes can occur. The rules governing possible states and state changes are termed *state laws* and *transition laws*, respectively.

In addition to the basic ontological concepts we define some more concepts to provide a formal basis for expressing process-related concepts in ontological terms.

Domain: Part of the world changes of which we want to model.

In ontological terms, a domain includes a set of things and their interactions. By defining a process over a domain we set the *scope* of the process and provide a clear distinction between what would be considered *external events*, which are outside of the process' control, and *internal events* that may occur while processes are enacted and are governed by the processes in the domain.

State: A set of time-dependent attributes (state variables) that provide sufficient information about the domain for the purpose of modeling.

Note, the state of the domain is determined by the states of the things included in it. However, due to interactions, emergent state variables of composite things or of the domain might exist. As well, we view states as being discrete, meaning that any change is from one state to another at a certain moment in time.

Sub-domain: Part of the domain that can be represented by a (fixed in time) subset of the set of state variables.

A state can be *projected* on a sub-domain by considering the sub-set of state variables describing the sub-domain. This subset defines the state of the sub-domain. Hence on "state" means a state of a domain or any sub-domain. A sub-domain may set the scope of a sub-process (part of the process occurring in the sub-domain), similarly to the way a domain sets the scope of a process.

States can be classified as being *stable* or *unstable.* The motivation to this distinction is that later on we will view the execution of a process as a sequence of unstable states that terminates when a stable state is reached.

Stable State: A state that can only change as a result of an action of something outside the (sub)domain.

Unstable state: A state of the (sub)domain that must change.

Whether a state is stable or unstable and how an unstable state might change is defined in terms of the *laws* that govern the states of the domain and their transitions:

A Law: A function from the set of states S into itself.

A Transition law: A function on the set of possible unstable states S_u into the set of states S.

Implied in this definition is that the transition law is fully deterministic. However, our process model allows for uncertainty in how the process will progress when enacted. This is because we allow for external events to affect the state of the domain while the process is in progress. Consequently, state variables that affect the law might change in ways not controlled in the process.

A transition law can be extended to all states as follows: for an unstable state the law is the transition law, otherwise it maps the state into itself. Hence on we will refer to the extended laws as domain laws (designated by L).

For modeling processes we will be interested in sequences of unstable states that terminate on stable states. It is not guaranteed that a domain law will always lead to a stable state. We therefore need a condition under which every process will terminate (i.e. the domain will reach a stable state).

Stability Condition: A domain will always achieve a stable state if for every $s \in S_u$ there exists n such that $L^n(s) = L^{n+1}(s)$.

The stability condition means that a stationary point exists for every sequence of unstable states. The sequence of unstable states, transforming by law until a stable state is reached is the basic mechanism of a process:

A Process: A sequence of unstable states leading to a stable state.

This definition of a process does not mention the origin of the initial unstable state. In particular, it can be the outcome of an interaction between the domain and a thing outside the domain when the domain is in a stable state. Furthermore, the definition does not mention the process goal explicitly. However, it implicitly assumes that the stability condition holds and a stable state can be reached.

Our purpose is to add the notion of goal to the formal model of a process. The formalization below establishes and operationalizes a process goal integrated into a process model. Note that by "goal" we relate to an operational goal of the process only, as opposed to business goals of the organization. Business goals, which may serve as "soft-goals" in process design, are discussed in [18].

Definition 1: Assume the set of state variables defining a domain is $s=<x_1,\ldots,x_n>$.

Let $S=\{s \mid s \text{ lawful}\}$ be the set of possible domain states. Let $S_{st} \subseteq S$ be the subset of domain stable states. Then a *Goal* (G) is a set of stable states $G \subseteq S_{st}$.

We now relate the notion of a goal to a process:

Definition 2: A goal G will be said to be a *process goal* if every execution of the process terminates in G.

Definition 2 is technical in the sense that it does not provide any meaningful understanding of how a process is designed to always end in a specific subset of states. We need to operationalize the concept of goal so it can be related to the actual design of a process leading to it.

Definition 3: A *criterion function* is a function on the set of states C: $S \rightarrow D$, where D is a certain domain (of values).

A criterion function maps the values of state variables into a domain where a decision can be made on whether the process achieved its purpose or not. Examples for criterion functions are the average of certain state variable values or their distance from a target value. Often, the mapping is on a subset of state variables that are considered relevant for deciding whether the process has reached its "goal". The domain mapped into is then a sub-domain of the process domain. For example, in a manufacturing process a criterion function can map the entire set of state variables into a sub-domain specified by two Boolean state variables sufficient for determining process termination: "Production is completed" and "Product quality is approved".

Definition 4: A *condition* is a logical expression E made of simple expressions of the form: R::= C rel g,

where rel∈{'>', '=', '<'}, where C a criterion and g is a value from the same domain as C, combined by 'AND', 'OR', 'NOT' and precedences indicated by '( )'.

We can now operationalize the definition of a goal as: G ={s | E(C(s)) is 'true'}.

Considering the manufacturing process discussed above, its goal set is
{s| (Production is completed='true') AND (Product quality is approved='true')}.

Having defined and operationalized the goal of a process, we can now apply a set of concepts to formalize a process model.

Definition 5: A *process model* is a quadruple M_P = <S,L,I,G> where:

S is a set of states
L is a law defined on S
I is a subset of unstable states in S: the set of possible initial states
G is a subset of stable states in S: the goal set

This definition does not seem to address many elements that are usually included in process models, such as ordered activities, pre and post conditions, resources, and actors. Nevertheless, we shall now show how the concepts used in the definition relate to those common process modeling concepts, by viewing a law as a mapping from a condition over a criterion function to a condition over a criterion function.

Ordered activities are state transitions caused by transformations defined by the law. *Triggering events* (or pre-conditions) are the conditions that define the set of initial states I of the entire process and sub-processes. *Post-conditions* are the conditions that define the goal sets for the process and sub-processes. *Actors and resources* are things in the ontological model. Our process model represents actors as things that take actions in response to their state changes and resources as things that take no further actions.

In summary, modelling the states that change by law from a set of initial states to a set of stable states is sufficient for addressing most of process model elements.

Note, definition 5 implies that a set of ordered activities is not a process unless it leads to a defined goal. Consequently, if L does not satisfy the stability condition a process cannot be defined.

3 Validity of Process Models

This section uses the GPM theoretical framework to develop conditions for identifying validity and completeness in a process model. Different sources of invalidity are indicated so that actions to remedy the invalidity can be suggested.

As discussed in Section 2, a process is aimed at attaining a goal, which is a set of stable states that satisfy a condition over a criterion function. Such a set can, potentially, be attained in many different ways or *paths*.

Definition 6: a *process path* is a set of states $<s_1,...s_n>$ such that: $s_i \neq s_j$, $s_1 \in I$, and $s_{k+1}=L(s_k)$ for every $k \in \{1,n-1\}$.

Note, definition 6 does not relate to the process goal. Accordingly, a process path might be unsuccessful by leading to a stable state which is not in the process goal set.

Definition 7: a *successful process path* is a path such that $s_n \in G$.

We base the evaluation of a process model on whether its goal is reachable or not. For this purpose, we define and discuss the reachability of states in the goal set.

Definition 8: A goal state $s \in G$ will be termed *reachable* if there is one (successful) process path such that $s_n = s$.

If the goal set of a process includes unreachable states it either means these states are redundant in the goal definition and the goal set can be redefined, or that some other process paths should be designed and successfully reach these states. However, we do not consider the existence of redundant goal states as model invalidity.

Definition 9: A process model will be called a *valid model* iff there exists at least one successful process path.

Three notes are in order. First, definition 9 relates to validity of a process with respect to a given goal. Second, the definition does not address the "validity" of the goal itself in terms of what the process is intended to accomplish. The result of a "faulty" goal definition may be a "valid" process model that does not provide all the outputs required from it (typically, by other processes). For example, assume at the end of a production process the identity of the worker is needed for computing salaries. Yet, providing this identity is not necessarily defined as part of the production process goal criterion function ("complete product"). Hence, the production process can be considered valid with respect to its goal set even if it does not provide the information required by other processes.

On a more general note - completeness of goal definition should be evaluated in relation to a set of processes. We do not discuss goal completeness here.

Third, a path might exist, yet not guaranteed to complete. The reason is that some events that bring about state changes might be considered external to the (sub) domain on which the process is defined. Since external events are not under the control of the process, the process might reach a stable state not in the goal that is not guaranteed to change further. For example, the domain of replenishment might include suppliers. However, for the replenishment process in the company, suppliers' behavior is external. Therefore order delivery is not an internal event of the process sub-domain. The process is not "guaranteed" to complete and might "hang" waiting for delivery.

Based on the above, we distinguish three types of situations when a process model is invalid, namely, it is not guaranteed to reach its goal. These types are (1) Incompleteness in the process definition. (2) Inconsistency between the law and the goal definition. (3) Dependency of the process on external events, where the process is "waiting" for an external event.

In what follows, we will discuss these situations and suggest remedies for the reasons the process model is not valid.

Incompleteness in the Process Definition: The domain and law definition determines which state variables are addressed by the model. Specifically, the domain law is defined in terms of a mapping between two conditions over criterion functions. It may be that a certain combination of state variable values obtained in a given step (according to the definition of the domain law), does not appear in the law definition for the following steps. As a result, the following step cannot be "fired".

Definition 10: A process definition is considered *complete* iff the domain law is defined for every combination of state variable values that may be reached from the initial process states via state transitions defined by the law.

Incompleteness in the process definition can relate to internal events or to external events. In the case of internal events not completely defined, it may be that steps 1..j of a process path lead to states defined by $C_1(x_1,..x_{n-1})$, while the initial state of step j+1 is defined by $C_2(x_1,..x_n)$. The law for the current value of x_n might not be defined. Consider, for example, a production-to-order process, in which production should be triggered by the acceptance of a customer order. Suppose that when an order is received not all details needed for production are provided. The law might not specify how to proceed. The problem can be solved simply by correcting the law (as defined for the order acceptance step) so that all the necessary details will be provided.

In the case of external events not completely defined, some state variables obtain values by external events and become known during the process (a formal definition of this situation appears in [18]). In words, the value of state variables that are determined by an external event is subject to uncertainty until it is *realized* during the process. Possibly, not all the potential results of the external event are taken into account by the law. Hence, the process might reach a state for which L is undefined.

As an example, consider a process of periodical car maintenance, which includes examining the state of various systems in the car (e.g., the braking system). If failure is detected it will be fixed, while otherwise no action is taken. The damage to be detected reflects an external event which has already occurred by the time the car is in the garage. It only becomes known during the maintenance process. The law defined for the maintenance process must consider all the possible states in the state space. This will in turn specify various possible process paths depending on the findings.

Inconsistency between the Law and the Process Definition: It is possible that as the process begins progressing, it reaches a state from which it cannot proceed further to reach a goal state. Two possibilities exit. First, the law keeps causing transitions without reaching a stable state. If the state space is finite, this would imply the process has entered an "infinite loop", meaning the law does not satisfy the stability condition. Second, it is possible the process has reached a stable state not in the goal for which there is no external event that can change it to an unstable state. For either case, this implies there is no continuous path from the start state to a goal state. In other words, the goal is inconsistent with the law. In practice, this situation can be recognized by detecting state variable values in the goal that can not be set properly by the law.

If the goal is derived from organizational needs, it should not be changed. Hence, the process model can be made consistent only by correcting the law definition.

Process continuance Depends on an External Event: An external event may be the trigger for a step in the process, in which case the process is in a stable state “waiting” for the external event to occur, with no guarantee that it will eventually occur.

Definition 11: A process path will be termed *non-continuous* iff it includes a stable state $s_j \notin G$.

Based on this definition, the only way for a non-continuous path to lead to the goal is if an external event occurs changing the stable state to unstable.

Definition 12: A process will be termed *non-continuous* iff all its paths are non-continuous.

A non-continuous process is technically invalid. However, it is common that a process will be waiting for external events (we do not want to proscribe this, as it would limit what is a useful process model). For example, handling a manuscript submitted for publication involves sending it to reviewers. Once the manuscript is sent to the reviewers the process is in a stable state, and is reactivated by the arrival of the reviews. However, this might take an indefinite period of time. In fact, theoretically there is no guarantee that all the reviews will arrive at all.

Assuring the process reaches its goal would require the following corrections:

(1) Add to the Law L a specification that will include some measure of time in the criterion function, and a condition specifying values in which the combination of the time variable and other state variables makes the state unstable. The time can be “absolute” or “waiting” time.

(2) The unstable state should be mapped to (a) connect to a process path, (b) end on a new stable state that can be affected by external events or be considered a new goal.

(3) Add the stable state of (2b) to the Goal set.

It can be shown that these corrections are all necessary and together sufficient. The idea is as follows: given the process might be halted on a stable state not in the goal set, the only way it can be reactivated is via an external event. In theory, this could be an event of any source. However, this might again raise the same possibility of failed external event. Thus, we must include a type of external event that by definition will always change the state. This must be time, as the passage of time is external to all processes in the world. In organizational practice one might conceive of other necessary events that would happen, however, they all will be somehow tied to time.

When defining a state that becomes unstable as a result of the time event, a new process path must be defined and connected to it. The new process path may include a *monitoring* activity, aimed at verifying that the expected external event will indeed occur. The monitoring activity itself might have to be tied to time.

In addition (or alternatively), the new process path may lead to an “exception” state, where a process terminates with some special actions reflecting a failure somewhere. The condition for reaching the exception state can be based on the time variable, on the number of repetitions of the monitoring activity (which in itself reflects time), or on a combination of both.

We distinguish between two types of stable states reached as a result of monitoring. First, the stable state is an exception and should be added to the goal set of the process, implying that the process terminates when its original objective is achieved or when failure in achieving this objective is evident. Second, the new stable state could be such that is more likely to be changed by an external event. For example, notify a supervisor of the problem, and wait for response.

Consider again the example of the manuscript handling process. In order to make it valid we need to specify time in which to inquire about the status of the review, or time periods for repeating this inquiry. We should also specify a path for failing to get a review, in which decision can be made based only on the reviews that have arrived.

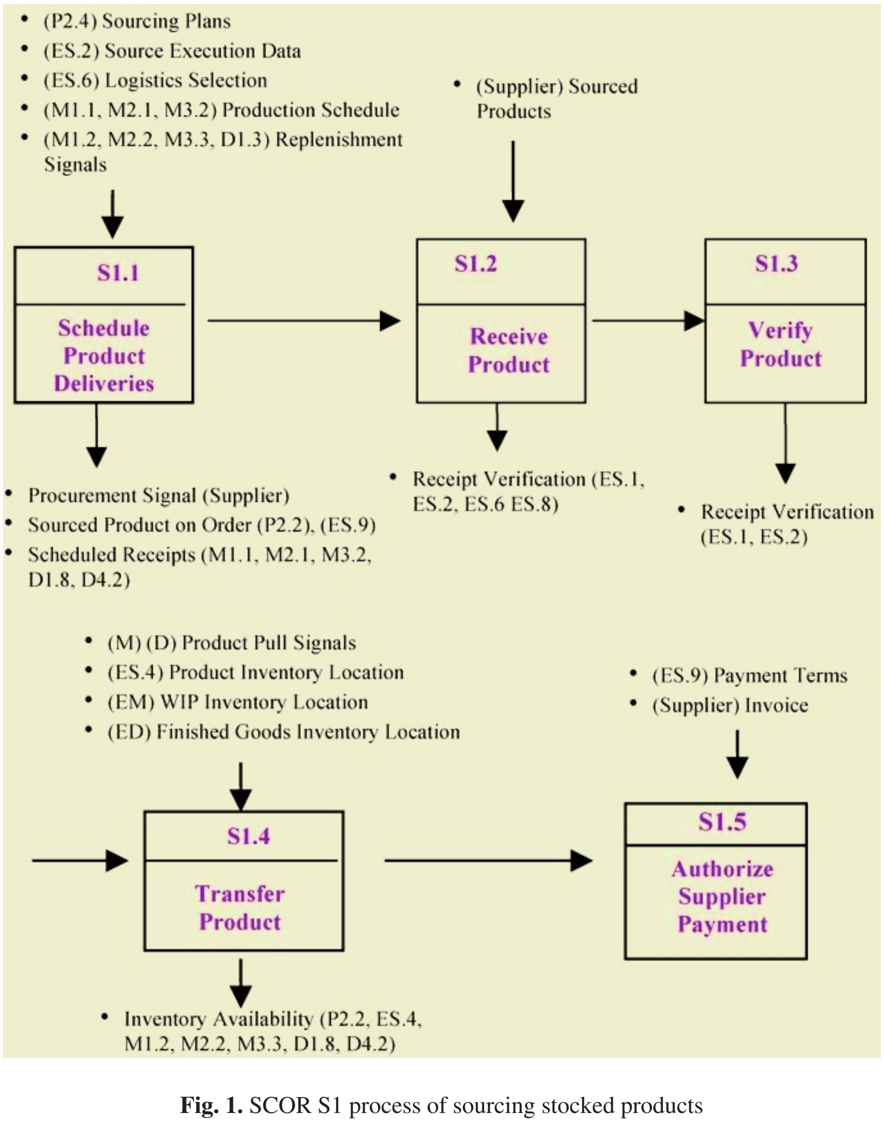
(P2.4) Sourcing Plans
(ES.2) Source Execution Data
(ES.6) Logistics Selection
(M1.1, M2.1, M3.2) Production Schedule
(M1.2, M2.2, M3.3, D1.3) Replenishment Signals
(Supplier) Sourced Products
S1.1
Schedule Product Deliveries
S1.2
Receive Product
S1.3
Verify Product
Procurement Signal (Supplier)
Sourced Product on Order (P2.2), (ES.9)
Scheduled Receipts (M1.1, M2.1, M3.2, D1.8, D4.2)
Receipt Verification (ES.1, ES.2, ES.6 ES.8)
Receipt Verification (ES.1, ES.2)
(M) (D) Product Pull Signals
(ES.4) Product Inventory Location
(EM) WIP Inventory Location
(ED) Finished Goods Inventory Location
(ES.9) Payment Terms
(Supplier) Invoice
S1.4
Transfer Product
S1.5
Authorize Supplier Payment
Inventory Availability (P2.2, ES.4, M1.2, M2.2, M3.3, D1.8, D4.2)

Fig. 1. SCOR S1 process of sourcing stocked products

4 Application to the SCOR Model

In this section we demonstrate our approach by applying it to a process taken from the Supply Chain Operations Reference-model (SCOR) [16].

The SCOR model is a reference model of supply chain management processes, developed and endorsed by the Supply Chain Council as a cross-industry standard. While primarily targeted for industrial use, the SCOR model has been used in quite a number of research works (e.g., [1, 8, 19]) as a comprehensive body of common and accepted supply chain business processes.

SCOR contains three levels of process details. The top level includes five basic processes: Plan, Source, Make, Deliver, and Return. The second level defines categories for each of the five basic processes, according to different logistic categories (e.g. make to stock). The third level decomposes each process category into elements, to be further decomposed into activities in practical implementation. The SCOR model specifies inputs and outputs of each of these elements, and provides metrics and "best practices" associated with each process category and element.

We demonstrate our approach using the SCOR process Source Stocked Products, denoted as S1, presented in Figure 1 (taken from SCOR directly). The inputs and outputs in the figure refer to other SCOR processes they relate to (e.g., P2.4, etc.).

4.1 Expressing the SCOR Process

The S1 process includes five steps (or sub-processes). We may apply our set of concepts to the entire process or to each of its steps, as shown in Table 1.

With respect to the entire process, I is the set of states where all the inputs to the first step exist. Specifically, the instability of the states is caused by the *replenishment Signals* given. Therefore, this is the external event that triggers the process. The Law is not completely and explicitly specified. Rather, it is manifested by the state transformation caused by each of the process steps. Table 1 specifies the conditions that define I and G for each step. The law is defined by the mapping from I states to G states. In the transformation from Figure 1 to Table 1 we applied domain knowledge to specify the exact condition values (e.g., *Sourcing plans* = In process), which were not provided in the SCOR model. The goal of the entire process is not specified in the SCOR model at all, but is implicitly understood as the set of states where supplier payment is completed. Considering each step, it is a sub-process defined over a sub-domain. Its goal states, while clearly not stable in terms of the entire domain, are stable in terms of the specific sub-domain. For example, consider the step *S1.2 Receive Product*, whose output is *Receipt Verification* (obviously with respect to quantity rather than quality). The goal of this step includes all the states where the quantity received is verified. It is clearly not a stable state in terms of the entire process, since the quality of the product is not yet verified. However, it is a stable state in terms of a sub-domain, which may include a person whose job is to unload goods and verify that the content of the shipment delivered. This person has completed his job once the quantity received is verified.

Table 1. SCOR process representation

Step	I condition	G condition
S1.1	(*Sourcing plans* = Open) AND (*Source Execution Data* = Defined) AND (*Logistics Selection* = Defined) AND (*Production Schedule* = Defined) AND (*Replenishment signals* = Given) AND (*Sourced products* = Not Ordered)	(*Sourcing plans* = In process) AND (*Replenishment signals* = Closed) AND (*Procurement signal* = Sent to supplier) AND (*Sourced products* = On Order) AND (*Scheduled Receipts* = Date, quantity)
S1.2	*Sourced products* = Arrived	*Sourced products* = Quantity verified
S1.3	*Sourced products* = Quantity verified	*Sourced products* = Quality verified
S1.4	(*Sourced products* = Quality verified) AND (*Pull signal* = Given) AND [(*Inventory location* = Defined) OR (*WIP location* = Defined) OR (*Finished goods location* = Defined)]	(*Sourced products* = Transferred) AND (*Pull signal* = Closed) AND [(*Inventory* = location, quantity) OR (*WIP* = location, quantity) OR (*Finished goods* = location, quantity)]
S1.5	(*Sourced products* = Transferred) AND (*Payment Terms* = Defined) AND (*Invoice* = Received)	(*Sourced products* = Paid) AND (*Invoice* = Paid)

4.2 Validity Analysis

Table 2 is a basis for analyzing the process model validity. The table addresses the state variables whose state is transformed by each step of the process and specifies the source from which each variable receives its initial and final value. The table also indicates whether a state variable constitutes (possibly in combination with other state variables) the external event that triggers the sub-process by putting the sub-domain in an unstable state. The sources of the initial values of the state variables can be external events or some other steps in the process. The source of the goal values of the state variables is the law, possibly on the basis of external events whose outcome is realized in the process (denoted in the table as By Law | external).

Note, the table specifies for each step only the *goal defining* state variables [18], i.e., state variables that are part of the goal criterion function of the specific step.

Analysis tables such as Table 2 can indicate the three cases of invalidity discussed:

(1) Incompleteness of process definition can be identified by tracking:
  (a) State variable whose initial value depends on a previous step of the process, where the required value is not specified as part of the goal condition of that step. Then a pre-condition might not be satisfied and a step cannot be fired. In our example the changes in *Sourced Products*, that progress from step to step, are specified correctly.
  (b) State variables whose goal value depends on external events (By law | external), where the law is not specified for all their possible values. In our example there are two such cases, namely steps S1.2 and S1.3. In both cases not all possible results are considered, as it is possible that the quantity will not be verified (S1.2) or quality will not be approved (S1.3). Hence, these are two cases of incomplete specification.

(2) Goal-law inconsistency can be identified by detecting loops in the process – when the initial value of state variables is obtained in the goal state of a step

that is not previous to the current one. Loops can also be the result of repeating external events. Such possibility can be identified if the initial state of a step depends on an external event only, without considering the value of goal-defining state variables (that changes once the step is performed). Once a loop is detected there must be a state variable whose value serves as a guaranteed termination condition (repetition counter, time, etc.). Otherwise the stability condition is violated. In our example the process does not include loops.

(3) Dependency on external events can be identified by state variables, which constitute triggering events for steps (other than the first step of the process), and whose initial value is of an external source. These could indicate that the process is non-continuous. In our example there are three such cases, in steps S1.2, S1.4, and S1.5. In all these three cases the occurrence of the external triggering events is not guaranteed and is not monitored.

In summary, Table 2 indicates that the process model is invalid, since it includes two cases of incomplete law specification with respect to state variables whose value is realized in the process, and three cases of unmonitored non-continuity.

Table 2. Sources of state variable values

Process step	**State variable**	**Source of initial value**	**Triggering event**	**Source of goal value**
S1.1	*Sourcing plans*	External	No	By Law
	Replenishment signals	External	Yes	By Law
	Procurement Signal		No	By Law
	Sourced Products		No	By Law
	Scheduled Receipts		No	By Law
S1.2	*Sourced Products*	External	Yes	By Law | External
S1.3	*Sourced Products*	S1.2	Yes	By Law | External
S1.4	*Pull Signals*	External	Yes	By Law
	Sourced Products	S1.3	Yes	By Law
	Inventory	External	No	By Law
	WIP	External	No	By Law
	Finished goods	External	No	By Law
S1.5	*Sourced Products*	S1.3	Yes	By law
	Supplier invoice	External	Yes	By law

4.3 Modifying the Process Model to Achieve Validity

In this section we suggest example corrective actions to achieve validity of the S1 process. Clearly, different solutions may be suggested, depending on the procedures of the specific organization. Table 3 summarizes the modified process model. The possible values of the realized state variables are specified and mapped to relevant goal states in S1.2 and S1.3. In S1.3 the case of unapproved quality will terminate the process, when the sourced products are to be returned to the supplier by another process and a re-planning signal is given. This state should be added to the goal of S1.

Table 3. Modifications to Process S1

Step	I condition	Realized value of state variables	G condition	Explanation
S1.1	See Table 1			
S1.2m	(*Time passed* = t_i) AND ($i \le n$) AND (*Sourced products* = Order Opened)		[(*Arrival Control* = OK) OR (*Arrival Control* = Exception)] AND ($i = i + 1$).	*i* is a counter. Begin monitoring if t_i time has passed.
S1.2e	($i > n$) OR (*Arrival Control* = Exception)		(*Sourced products* = Order closed) AND (*Re-planning signal* = given).	If monitoring time ended ($i > n$) or exception is recognized by inquiry – process ends
S1.2	*Sourced products* = Arrived	*Actual quantity* = *Stated Quantity*	(*Sourced products* = Quantity verified)	
		Actual quantity ≠ *Stated Quantity*	(*Sourced products* = Quantity verified) AND (*Claim to supplier* = to be sent)	
S1.3	*Sourced products* = Quantity verified	*Quality* = meets specification	*Sourced products* = Quality verified	
		Quality ≠ meets specification	(*Sourced products* = to be returned) AND (*Re-planning signal* = given)	
S1.4m	*Pull signal* ≠ Given		*Pull Control* = OK	Monitoring by an immediate notification
S1.4	See Table 1			
S1.5m	(*Time condition* = t) AND (*Invoice* ≠ Received)		*Invoice Control* = OK	Monitoring by notification when waiting time = t
S1.5	See Table 1			

The non-continuous parts need, as discussed in Section 3, to be monitored. The monitoring procedure may vary, depending on the level of uncertainty related to the external event and on the criticality of continuance to the organization. In the case of S1.2, waiting for goods to arrive from the supplier involves a relatively high level of uncertainty and is indeed critical. Hence we proposed a time-dependent control, which repeats itself in time intervals of t_i, where i is the number of repetitions. The exception path will be taken based on the number of monitoring repetitions or on the inquiry response, and its goal state should be added to the goal set of the entire process, S1. The monitoring activity is denoted in Table 3 as S1.2m, and the exception, which terminates the process, is marked S1.2e.

The two other cases of discontinuity in the process (S1.4, S1.5) seem less critical or likely to occur. Hence, the monitoring suggested (S1.4m, S1.5m) is less tight as it includes no repetitions, and a possible exception is not considered.

This demonstrates how the proposed model can be used as the basis for design decisions and the type of considerations that may guide the process' designer.

5 Related Work

Related work includes the areas of goal-driven process models and process model verification. Attempts to incorporate goals into process modeling include [12], who suggest an informal approach in which goals provide a basis for process definition. Business process modeling is addressed by [10] using the Enterprise Knowledge Development (EKD) framework, which entails a goal model among other views, and sets the understanding of goals as a basis for business process identification.

A formally defined set of concepts, incorporating goals and processes, is provided in [11], whose model is based on mathematical systems theory. Their approach to process modeling is state-oriented, viewing a process as a subset of trajectories in some state space, and a process goal as a set of conditions defining a surface in the state space. This set of concepts is extended in [2] and used for defining a process pattern, allowing the design of generic processes that can be specialized for specific situations. This model bears much similarity to our model. However, the distinction of external events is not explicitly made there.

Verification of process models is mainly associated with workflow control models. Workflow model verification is notation-specific, defined often for Petri-nets [20, 21], and sometimes for other languages, e.g., UML Activity diagrams [6].

Basically there are two approaches to the verification. In one, the model is converted into formal specifications that can be analyzed by existing formal model checkers (e.g., SPIN, that also served for verifying UML statecharts [5, 13]) or dedicated model checkers [6]. The other approach is based structural properties of the model (e.g., soundness) [20, 21]. Note, our validity criteria are not structural only, addressing semantics as well via the values of the state variables in the model.

Our process model is less restrictive than the soundness property required in workflow models. For example, in sound workflow models only one termination place is allowed, in contrast to our goal set.

Note, workflow models represent the behavior of the workflow management system (WFMS) only and not its environment. Specifically, workflow processes are usually non-continuous, since the WFMS has to wait for human actions to be reported to it. Hence, workflow models generally assume the environment behaves fairly [21]. In addition, as opposed to workflow verification methods, our approach is conceptual rather than technical. It explores the sources of invalidity and provides remedy to specific cases. Finally, it is independent of any specific notation.

6 Concluding Discussion

The GPM, proposed in this paper, is a theory-based model of a process. This model is utilized for defining validity of a process model and identifying causes for invalidity. Understanding these causes can lead to suggestions for correcting invalid models.

The suggested process model is goal-driven, basing the notion of validity on goal reachability. A process goal is not an obscure notion, but a set of stable states defined by a condition. Hence, goal reachability can be systematically verified.

The verification criteria, though systematic, are conceptual rather than technical. They provide an understanding of the sources of invalidity, so the insight gained will assist the modeler in designing valid process models from the beginning. Alternatively, they provide guidance for modifying the model and make it a valid one.

The suggested process model is generic and notation-independent. It employs a small number of constructs to express many aspects addressed by various process modeling languages. Consequently, models in these languages can be mapped to our generic model and their validity evaluated regardless of the specific notation.

Furthermore, applying the suggested model as an infrastructure to models created in any modeling language can help to structure the modeling process, by presenting a set of questions to the modeler. As an example, assume a process is modeled using Petri nets. Normally the modeler would be occupied with transitions (activities) and firing sequence. Using our model as infrastructure, the modeler will also have to understand the process goal (a condition over a criterion function) and define the places in the model in terms of state variables.

The application of the GPM framework is not limited to validity evaluation. Currently we are extending it to other aspects of process modeling, such as process decomposition, process specialization, and process model reuse.

References

1. Arns, M., Fischer, M., Kemper, P., and Tepper, C. (2002), "Supply Chain Modelling and its Analytical Evaluation", Journal of the Operational Research Society, Vol. 53, pp. 885-894.
2. Bider, I., Johannesson, P., Perjons, E. (2002), "Goal-Oriented Patterns for Business Processes", Position paper for Workshop on Goal-Oriented Business Process Modeling (GBPM'02).
3. Bunge. M., *Treatise on Basic Philosophy: Vol. 3, Ontology I: The Furniture of the World.* Reidel, Boston, 1977.
4. Bunge. M., *Treatise on Basic Philosophy: Vol. 4, Ontology II: A World of Systems*, Reidel, Boston, 1979.
5. Eshuis, R., Jansen, D. N., and Weiringa, R. (2002), "Requirements-Level Semantics and model Checking of Object-Oriented Statecharts", *Requirements Engineering*, 7, pp. 243-263.
6. Eshuis, r., and Weiringa, R. (2002), "Verification Support for Workflow Design with UML Activity Graphs", *Proceedings of the 24th International Conference on Software Engineering (ICSE)*, ACM Press NY USA, pp. 166-176.
7. Hammer, M. and Champy, J. (1994), *Reengineering the Corporation – A manifesto for Business Revolution*, Nicholas Brealey Publishing, London.
8. Humphreys, P. K., Lai, M. K., and Sculli, D. (2001), "An Inter-organizational Information System for Supply Chain Management**",** International Journal of Production Economics, Vol. 70 No. 3, pp. 245-55.
9. Johannesson, P. and Perjons, E., 2001, "Design Principles for Process Modeling in Enterprise Application Integration", *Information Systems* 26 pp. 165-184.
10. Kavakli, V., and Loucopoulos, P. (1998), "Goal-Driven Business Process analysis Application in Electricity Deregulation", in Pernici, B. and Thanos, C. (ed.), Advanced Information Systems Engineering (CAiSE'98), LNCS 1413, Springer-Verlag Berlin, pp. 305-324.

11. Khomyakov M., and Bider, I. (2000), “Achieving Workflow Flexibility through Taming the Chaos” OOIS 2000 - 6th international conference on object oriented information systems. Springer-Verlag Berlin, pp. 85-92.
12. Kueng, P., and Kawalek, P. (1997), “Goal-based Business Process Models: Creation and Evaluation”, BPMJ, Vol. 3 No.1, pp. 17-38.
13. Latella, D., Majzik, I., and Massink, M. (1999), Automatic Verification of a Behavioural Subset of UML Statechart Diagrams Using the Spin Model-checker, *Formal Aspects of Computing*, 11, pp. 637-664.
14. Paulson, D. and Wand, Y., (1992) “An Automated Approach to Information Systems Decomposition”, IEEE Transactions on Software Engineering, Vol. 18 No. 3, pp. 174-189.
15. Scheer, A. W., 1999, *ARIS – Business Process Frameworks*, Springer, Berlin.
16. SCOR Reference model, Supply chain council. www.supply-chain.org.
17. Soffer, P., Golany, B., Dori, D., and Wand, Y. (2001) “Modeling Off-the-Shelf Information Systems Requirements: An Ontological Approach”, Requirements Engineering, Vol. 6, pp.183-199.
18. Soffer, P., and Wand, Y., 2003, “On the Notion of Soft Goals in Business Process Modeling”, *Business Process Management Journal* (to appear).
19. Stephens S. (2001), “Supply Chain Operations Reference Model Version 5.0: a New Tool to Improve Supply Chain Efficiency and Achieve Best Practice”, Information Systems Frontiers, Vol. 3 No. 4, pp. 471-476.
20. Van der Aalst, W. M. P. (1997), “Verification of Workflow Nets”, *Application and Theory of Petri Nets*, LNCS 1248, Springer-Verlag, Berlin, pp. 407-426.
21. Van der Aalst, W. M. P. And Ter Hofstede, A. H. M., 2000, “Verfication of Workflow Task Structure: A Petri-net-based Approach”, *Information Systems* 25(1), pp. 43-69.
22. Wand, Y. and. Weber, R (1990), “An Ontological Model of an Information System”, IEEE Trans. on Software Engineering, Vol. 16, No. 11, pp. 1282-1292.