Goal Orchestrations: Modelling and Mining Flexible Business Processes

By Metta Santiputri

Goal Orchestrations: Modelling and Mining Flexible Business Processes

Metta Santipuri[1], Aditya Ghose[1(✉)], Hoa Khanh Dam[1], and Suman Roy[2]

[1] Decision Systems Lab, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia
{ms804,aditya,hoa}@uow.edu.au

[2] Infosys Ltd., #44 Electronics City, Hosur Road, Bangalore 560 100, India
Suman_Roy@infosys.com

Abstract. In many application domains, it is more natural to think of a process as a coordination model of goals to be achieved rather than of tasks or acitivities to be performed. Replacing tasks or activities with goals in process models allows us to enact processes in flexible, context-sensitive ways. We define a formal semantics for processes modeled in this manner (which we call *goal orchestrations*) and show how these enable flexible process execution. We also offer a simple means of mining goal orchestrations from readily available event logs, and present an evaluation with an event log consisting of 65000 entries from one of the world's largest IT companies.

1 Introduction

Business processes (and related conceptions, such as clinical processes or manufacturing processes) are typically specified in terms of *tasks that need to be executed*. This ignores an important alternative perspective on process modeling, in terms of *goals to be achieved*. Consider a physician preparing a patient for surgery. The treatment plan is typically conceived of as a sequence of goals to be achieved: *first we will lower the patient's blood pressure, then we will stabilize the patient's blood glucose levels, then treat the persistent chest infection before sending the patient into surgery*. Implicit in this treatment plan is a sequence of three goals to be achieved: "lower blood pressure", "stabilize blood glucose" and "treat chest infection". It is interesting to note (and this has been borne out by extensive interviews with physicians) that treatment plans do not involve task descriptions such as "administer drug X" or "treat with antibiotic Y". The key motivation behind conceiving these treatment plans as sequences of clinical goals/objectives to be achieved is to admit the possibility of achieving these goals in multiple different ways. Indeed clinicians often flesh out additional detail in such treatment plans by introducing what might be viewed as sub-processes of the form: *if medication X (for blood pressure) does not work, we'll try medication Y concurrent with hydration therapy*. Similar patterns of modeling can be

Suman Roy did this work when he was a visiting fellow at University of Wollongong on Infosys-CRC funded project of data-driven process discovery during July–Dec'14.

H.C. Mayr et al. (Eds.): ER 2017, LNCS 10650, pp. 373–387, 2017.
https://doi.org/10.1007/978-3-319-69904-2_29

found in a range of other settings (manufacturing, logistics as well as traditional "business" process application domains).

Our objective in this paper is to formalize this alternative approach to process modeling via *goal orchestrations*. As the name suggests, a goal orchestration is a coordination or orchestration model, where we describe the coordination of goals instead of the coordination of tasks. There are several reasons why these are of interest. First, goal orchestrations provide a more natural means of modeling behaviour (or processes) in many settings, as illustrated by the clinical example above. Second, goal orchestrations provide an easy means of achieving flexible process execution. The execution machinery for goal orchestrations is able to compute alternative task-level realizations of a goal if the initial attempt at realizing the goal fails to achieve the desired results (manifested by *events* or *effects* in the operating context). Third, goal orchestrations offer abstract, strategy-level views on processes, which can aid human understanding and ease process redesign (and other forms of analysis). More interestingly, as we will show below, goal orchestrations can be mined from readily available enterprise data in the form of *event logs*.

Goal orchestrations and their analysis involves considerable complexity (even though a superficial reading might suggest that all we are doing is replacing tasks in process models with goals). We need to consider temporal coordination of entities at three, progressively finer-grained, levels of abstraction: (1) the *goal level*, (2) the *task level* and (3) the *event level*. A given goal orchestration might admit multiple *conformant* task sequences. We need to use an important device from the reasoning about action literature, in the form of a *state update operator* to accumulate the persistent effects of tasks at the event level. The fact that state update, in general, leads to multiple non-deterministic outcomes means that the execution of a given task sequence might generate multiple event sequences (the non-determinism implies that we will be unable to specify at design time which of these event sequences will actually accrue when the task sequence is executed).

Of special interest are goal orchestrations that leverage an input *goal model* (in the form of an AND-OR goal graph). The input goal model can help identify alternative OR-refinements of a given goal, opening up a larger space of re-design alternatives in settings where the run-time monitoring machinery detects that the execution of a goal orchestration has not delivered the desired effects. The AND-refinements of a goal specified in a goal model can also be leveraged, but we do not formalize this in detail in this paper due to space constraints.

We note that much of the machinery we describe requires repeated use of the state update operator, and entailment checks. While space precludes a detailed empirical evaluation of computational cost of these, we can note that in the case of a propositional language for describing events/effects, fast SAT-solvers can execute these checks in near real-time. Well-known results about Horn clause theories also indicate that entailment checking can be executed in polynomial time.

In this paper, we provide a formal semantics of a goal orchestration model (in Sect. 2) by using abstractions spanning the goal level, the task level and finally the event level. We outline a machinery for executing goal orchestration models

that achieves the flexibility discussed above (Sect. 3). We then outline an approach to mining goal orchestration models (Sect. 4) from event logs, leveraging in considerable detail the semantics described in Sect. 2. Finally, we present an empirical evaluation (Sect. 5), first with a synthetic dataset, and then with a dataset from one of the world's largest IT companies involving an event log with 65,000 distinct entries.

2 Goal Orchestration Models and Semantics

A goal can be represented in any truth-functional language that comes equipped with machinery for checking satisfiability (and hence entailment). In the following, we will only consider *achievement goals*. A *goal orchestration* (N, F) is best viewed as a process graph (as commonly used in the literature) with the tasks/activities replaced with goals, where $N = G \cup \Gamma \cup E$ (G is a set of goal assertions, Γ is a set of gateways, and $E = E_s \cup E_f$ is a set of special events (E_s represents start events and E_f denotes end events); $F \subseteq (N \backslash E \times N \backslash E) \bigcup (E_s \times N \backslash E) \bigcup (N \backslash E \times E_f)$ corresponds to sequence flows connecting goal assertions with goal assertions, goal assertions with gateways, gateways with goal assertions, start events with goal assertions and goal assertions with end events. We will now describe the semantics by specifying under what circumstances an *event log* will be deemed to *satisfy* a goal orchestration. Recall that an event log is a set of pairs of the form $\langle event, timestamp \rangle$ (we ignore case IDs in the formulation, but if they are available, we can leverage these to cluster effects by process instance, if that granularity of analysis is of interest). We order an event log from the earliest timestamp to the latest, obtaining a sequence $\langle e_1, e_2, \ldots, e_n \rangle$, where each element is of the form $e_i = \langle \epsilon_i, \tau_i \rangle$ (ϵ_i is the i-th event, τ_i is its timestamp) and for every adjacent pair of elements in the sequence $\langle e_i, e_{i+1} \rangle$, $\tau_i \leq \tau_{i+1}$.

Every event involves one or more state transitions (a business object such as an insurance claim transitions from a *not-determined* state to an *accepted* state, or a task object transitions from an *incomplete* state to a *completed* state etc.). The effects of some events *persist* (an insurance claim once accepted remains in the accepted state) while others do not (a light that is initially switched on is eventually switched off). An event log describes the changes but not the non-changes. In other words, such a log describes new events as they occur but does not describe which prior events have persistent effects, so determining which effects hold at a given point requires specialized machinery. In the following, we will not distinguish between an event and its effects - thus the description of an event is also the description of its effects. To obtain a sequence of states or partial states (each denoted by a conjunction of *effect assertions*) from an event log, we *accumulate* effects using a *state update operator* in a manner similar to the approach adopted in [20,32]. A state update operator takes a state description and the effects of an action to generate one or more descriptions of the state that would accrue from executing this action in the input state. Some well-known state update operators are the Possible Worlds Approach (PWA) [15] and the

Possible Models Approach (PMA) [33] (other approaches based on the logic of theory change [3,12] and belief merging [23,24] can also apply, but we defer that discussion to future work). Given a set of accumulated effects (representing a possibly partial description of the state of the operating environment), and a new effect (representing the action just performed), we use the state update operator to determine what new set of accumulated effects should be (in our evaluation, we use the PWA operator, but others could be used without loss of generality). Applying the state update operator (denoted by $\oplus$) leads to non-deterministic outcomes. Thus, if s_1 and s_2 are states represented as conjunctions of event/effect assertions (we can think of the effects of an action being described, without loss of generality, as s_2), then $s_1 \oplus s_2$ is a set of states (the intuition being that any one of these could be the result of making s_2 true in state s_1.

The idea now, is to generate from an event log a sequence of *sets of states* (we need sets of states and not single states because of the non-deterministic nature of the state update operator). Given a set of prior states and a set of posterior states (i.e., those obtained from the prior set via state update), it is important to note that a state in the posterior set can be arrived at only from some (but possibly not all) of the states in the prior set. Thus, there are predecessor-successor relationships connecting elements of temporally adjacent sets of states. We first extract from an event log a *state set sequence* consisting of pairs of states, where the first element is the predecessor and the second element is the successor. Given an event log $\langle e_1, e_2 \ldots, e_n \rangle$, we compute a *state set sequence* $\langle StateSet_1, StateSet_2, \ldots, StateSet_n \rangle$, where each $StateSet_i$ is of the form $\{StatePair_1, StatePair_2, \ldots, StatePair_k\}$ and each $StatePair_i$ is of the form $\langle state_{pred}, state_{succ} \rangle$ (i.e., these are predecesor-successor pairs) as follows:

- We set $StateSet_1 = \{\langle \emptyset, \epsilon_1 \rangle\}$ (where $\langle \epsilon_1, \tau_1 \rangle$ is the first entry in the temporally ordered event log).
- We set $StateSet_2 = \{\langle \epsilon_1, s \rangle \mid s \in \epsilon_1 \oplus \epsilon_2\}$ (where $\langle \epsilon_2, \tau_2 \rangle$ is the first entry in the temporally ordered event log).
- For $i = 3 \ldots n, StateSet_i = \{\langle s_{i-1}, s_i \rangle \mid s_{i-1} \in StateSet_{i-1}$ and $s_i \in s_{i-1} \oplus \epsilon_i\}$.

A *state sequence* $\langle s_1, s_2, \ldots, s_n \rangle$ is supported by a state set sequence $\langle StateSet_1, StateSet_2, \ldots, StateSet_n \rangle$, if and only if:

- $StateSet_1 = \{\langle \emptyset, s_1 \rangle\}$.
- Every adjacent pair $\langle s_{i-1}, s_i \rangle$ in the state sequence must be an element of $StateSet_i$ in the corresponding state set sequence.

Given a state sequence $\langle s_1, s_2, \ldots, s_n \rangle$ and a goal model with a goal set $\{g_1, g_2, \ldots, g_k\}$ (this represents our vocabulary of goals), we compute a *goal sequence* $\langle G_1, G_2, \ldots, G_n \rangle$ by setting each $G_i = \{g_i \mid s_i \models g_i\}$. Note that a goal sequence is a sequence of sets of goals. We define a *goal orchestration trace* as a sequence of goals $\langle g_1, \ldots, g_m \rangle$ satisfying the constraints of the corresponding goal orchestration model (much like a trace through a process model). Given a goal orhestration model and a trace $\langle g_1, \ldots, g_m \rangle$, we will say that the trace is *supported by* a goal sequence $\langle G_1, G_2, \ldots, G_n \rangle$ if it is the case that $n \geq m$ and every $g_i \in G_i$.

Given a goal model (and thence, the set of goals contained in it), an event log *satisfies* a goal orchestration model if and only if a goal sequence can be obtained from the event log and the goal model in the manner described above such that the goal sequence supports a trace for the goal orchestration model.

3 Executing Goal Orchestrations

For a goal orchestration approach to enable flexible process execution, we require tasks/activities or enterprise capabilities to be annotated with post-conditions, specified in the same ontology as the goals (as recent results in [27] show, post-conditions can be relatively reliably mined from readily available enterprise data). More generally, one can view this as an instance of a generic scheme that permits us to relate task execution to the functional outcomes that are used to specify goals. A number of recent proposals suggest that leveraging task post-condition annotations can be effective and practical [6–8,10,11,13,19,20,29,32].

The first question we need to address is whether a goal orchestration is *feasible* with respect to an *enterprise capability library*. We shall view the latter as a repertoire of tasks or capabilities annotated with post-conditions. A goal orchestration is *strongly feasible* with respect to an enterprise capability library if and only if for every trace admitted by the orchestration, there exists a task/capability sequence $\langle t_1, \ldots, t_n \rangle$ with a corresponding sequence of post-conditions $\langle p_1, \ldots, p_n \rangle$ such that this latter, if viewed as an event log (this can be easily done by inserting time-stamps with each post-condition that respects the relative ordering), generates (given a goal model) a goal sequence that *supports* that trace. In the case of *weak feasibility*, we only require that there exist a task sequence that generates a goal sequence that supports at least one trace. The subsequent analyses will only be performed for goal orchestrations that are (strongly or weakly) feasible with respect to the available enterprise capability library.

Practical deployment of goal orchestrations must ideally be done with a goal model at hand. A goal model, typically an AND-OR goal tree, is critical in offering alternative means of arriving at the same outcome. We will refer to any goal related to a parent goal g in the goal model via an OR-link as an OR-refined child goal, and the OR-refined children of these and so on as the OR-refined descendants of g. We shall refer to the set of all OR-refined descendants of a goal g as the *OR-alternatives of g*. Given a goal orchestration model *GOM*, the set *OR-Alt(GOM)* of OR-alternatives of *GOM* consists of all goal orchestration models obtained by replacing at least one goal in *GOM* with an OR-alternative.

Executing a goal orchestration model consists of computing an *optimal suffix* for a partially executed task sequence (empty at the start of execution). By introducing a *current state* into the problem, one can deal with the problem of *semantic compensation* [17], where a process deviates from the functionality it is expected to deliver (manifested via events/effects) and where the challenge is to compute a new sequence of activities that will restore the process to *semantic conformance* (where it delivers the expected effects) and achieve the final goals.

Formally, given: (1) The current state S of the process and its environment, (2) a goal orchestration model and (3) The current sequence of goals achieved $\langle g_1, \ldots, g_i \rangle$, compute: a sequence of tasks $\langle t_j, \ldots, t_m \rangle$ drawn from the enterprise capability library such that the corresponding sequence of task post-conditions $\langle p_j, \ldots, p_m \rangle$, when concatenated with the achieved goal sequence $\langle g_1, \ldots, g_i \rangle$ generates a sequence of events $\langle g_1, \ldots, g_i, p_j, \ldots, p_m \rangle$ which can be viewed as an event log (with the appropriate insertion of sequence-maintaining time-stamps, as before) that generates a goal sequence that supports a goal trace through the input goal orchestration model.

Goal orchestrations serve to provide useful abstractions of underlying process models. Figure 2 shows a goal orchestration model for treating head injuries, providing an abstract view of a more detailed clinical process model in Fig. 1. A comparison of the 2 models reveals that the goal of administering IV bolus of dextrose is to maintain blood glucose level within the normal range, while giving extra dextrose helps achieve the goal of body fluid balance, and so on. 'Administer Paracetamol', 'Administer a bolus of IV morphine (50–100 µg/kg) and a morphine infusion (20–40 µg/kg/hr)', and 'Sedation' tasks are alternative ways of achieving the goal *Reduce patient's pain and stress*.

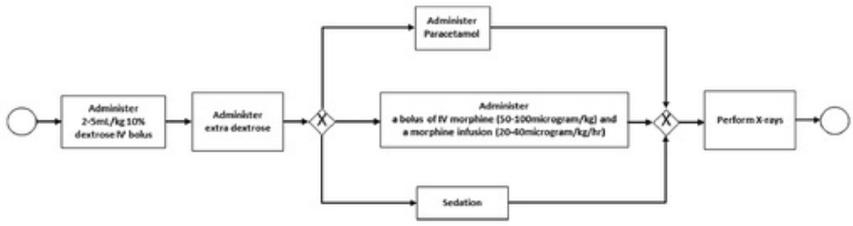

Fig. 1. Clinical process model fragment for treating head injuries

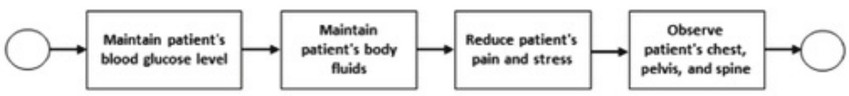

Fig. 2. Goal orchestrations for clinical process model in Fig. 1

4 Mining Goal Orchestrations

In this section, we show how goal orchestrations can be mined from event logs. A formal statement of the problem is as follows. Given: (1) An event log and (2) a goal model, compute: a goal orchestration that best explains the behaviour encoded in the event log. Recall that an event log records two kinds of events: events that flag the execution of a task and events that describe state transitions in objects impacted by a process. Our interest is in the latter kind of event (we shall refer to these as *effects*). It is useful to note that we do not need case IDs

associated with effects. Given a set of effects, we are only interested in their temporal ordering, but not which process instance, or actor/agent, might have generated these effects. Our intent is to identify the sequence of goals achieved (and thence a goal orchestration model) from the sequence of effects manifested. The vocabulary of available goals (as provided in the input goal model) provides the lens through which we view the effects. If the goal model is specific to an actor or a process instance, then the goals we will recognize and mine will be specific to the process or actor in question.

Mining goal orchestrations from event logs involves a sequence of pre-processing steps, followed by the application of an off-the-shelf process mining tool (in the empirical evaluation presented in the next section, we use AlphaMiner from the ProM toolkit [31]). The steps involved are as follows:

- Processing an event log to obtain a *state set sequence*.
- Extracting a set of *state sequences* from the state set sequence.
- Extracting *goal sequences* from the state sequences.
- Extracting a set of ordering assertions from each goal sequence identified in the previous step (we do not elaborate this in any greater detail since this is the standard approach associated with the AlphaMiner tool).
- Running an off-the-shelf process mining tool (AlphaMiner) with the goals playing the role of tasks.

5 Evaluation

The purpose of the evaluation is to establish that our approach is capable of the following:

- Mining goal orchestrations from readily available data
- Identifying different alternatives to achieving a goal based on the execution history

We present two cases to perform our evaluation. The first case involve a synthetic dataset and the second evaluation using a real-life dataset from a ticket handling process.

Evaluation with Synthetic Process Models: We ran the first experiment with a synthetic semantically annotated process model (i.e., a process model where each task is associated with the events/effects that would be generated as a consequence of executing that task) using $T_1, T_2, \ldots$ etc., for task names and $p, q, \ldots$ for effects. The model consists of 12 tasks with an XOR-split leading to two alternative flows, one of which included a nested AND-split and the other a nested XOR-split. The semantic annotations were 2 or 3 literals long and involved a mix of conjunctions and disjunctions. We generated a large number of possible execution traces of this model, and obtained the synthetic log using BIMP (The Business Process Simulator)[1] (with a small process model, performing the execution by hand also produced similar logs). We also investigated the

[1] http://bimp.cs.ut.ee/.

effect of scaling up the complexity of the process model, by generating a second synthetic process model with 20 tasks with and XOR-split leading to four alternative flows, one flow included a nested AND-split, two included XOR-split (one leading to two alternative flows and the other leading to three alternative flows), and the other was a sequence.

We randomly assigned effects to tasks, then performed the pre-processing steps described in the previous sections to obtain goal sequences, and from there, mined the goal orchestration. We had access to the ground truth (by maintaining the original process models together with the effects associated with each task and the goal sequence of each trace in the process model) so that we were able to determine the fitness and precision values for the mined goal orchestration.

Table 1 below describes the results of the experiments with each of these two process models. We measure the fitness and precision of the goal orchestrations generated from the log. Fitness evaluates whether the observed process complies with the control flow specified by the process, while precision indicates how precisely the model describes the observed process. In both process model, the results shows that the goal orchestrations generated from the mining conform to the data. The results appear overly predictable, but serve to establish feasibility and provide a baseline.

Table 1. Evaluation result with synthetic data

# of instances	# of events	Fitness	Precision	Time (ms)
Process model 1				
100	1520	1.0	1.0	52
500	7540	1.0	1.0	160
1000	15094	1.0	1.0	257
5000	75640	1.0	1.0	548
10000	151080	0.99	1.0	1,149
Process model 2				
100	1810	1.0	1.0	95
500	9008	1.0	1.0	287
1000	18026	1.0	1.0	377
5000	90040	0.99	1.0	1,170
10000	180540	0.99	1.0	3,147

The synthetic effect logs used in these examples considered all possible flows. Real-life data might involve more imperfections (such as certain XOR flows never being executed, certain tasks never being executed and so on).

Evaluation with a Real-life Dataset: An important part of the evaluation of the feasibility of the overall approach to goal orchestration was to gain experience

in using it in with a real-life dataset in a large complex practical setting. Our intent was to test several key elements of our proposal, including the processing (and pre-processing) of event logs, the identification of goals and goal sequences and the eventual use of process mining to obtain explicit goal orchestration models. Specifically, we looked at data from a team in one of the world's largest IT companies that supports IT infrastructure management as an outsourced service. Much of its activities involves the handling and resolution of *problem tickets* generated by customers. These can span the spectrum of complexity from a simple password reset to dealing with a complete ATM network that might have gone down. The dataset we analyzed described how 65000 distinct problem tickets were handled.

In the ticket handling process, when a member of a client firm faces IT-related problems or has queries about the IT systems whose management has been outsourced, they raise a ticket. The ticket handling system maintains records of ticket status from the opening of a ticket until the closing of it, responds with an acknowledgment to the user along with a notification to a system engineer who is assigned to handle the ticket. Also further input from the user may be requested. At this stage, if the problem can be resolved, the ticket is closed. In case the problem can not be resolved, the system checks to see whether there is any update from the user. If no update is provided and the ticket is not re-opened within a stipulated time, then the problem is considered as resolved and it is automatically closed. If the ticket is updated with new information then the system checks the nature of the ticket, whether it is incident or request, depending on which the ticket is serviced or resolved respectively.

The system records all events related to a ticket in the process. Each record represents all attributes of a ticket, such as incident number or ticket number to identify any particular ticket, the identity of the user or employee that raised the ticket, the timestamp of when the ticket is raised (`open date` attribute), when the problem is resolved (`resolve date` attribute), when the system sends a response to the user and the engineer (`respond date` attribute), when the ticket is closed (`close date` attribute), an attribute to signify if the ticket is reopened, etc. These attributes will be used to identify the current state of the ticket. For example, a ticket in the `Open` state signifies that the ticket has been received and currently at the start of the ticket handling process. Similarly, a ticket in `Close` or `Auto-close` state signifies that it is at the end of the process, etc. Based on these timestamps, we were able to identify 16 distinct event sequences, shown in Table 2.

We use the goal assertions in Table 3 to recognize goal sequences from event sequences (these goal assertions were provided by domain experts from the organization - the authors might have articulated these goals somewhat differently).

We extract a goal sequence for each event sequence in Table 2. Recall that a goal is recognized in an event if the formal representation of the event entails the formal assertion of the goal. The complete list of goal sequences thus obtained is presented in Table 4.

Table 2. Effect sequences identified in the log

Event sequence name	Event sequence	# of sequences
TR1	{open}, {open,respond}	1299
TR2	{open}, {open,respond,¬receive}	4546
TR3	{open}, {open,respond}, {open,respond,close}	2
TR4	{open}, {open,respond}, {open,respond,resolve}, {open,respond,resolve,close}	53296
TR5	{open}, {open,resolve}, {open,resolve,auto-close}	128
TR6	{open}, {open,approved}	70
TR7	{open}, {open,respond}, {open,respond,receive}, {open,respond,receive,¬reopen,auto-close}	25
TR8	{open}	296
TR9	{open}, {open,¬approved}	383
TR10	{open}, {open,respond}, {open,respond,rejected}, {open,respond,rejected,close}	1
TR11	{open}, {open,respond}, {open,respond,reopen,auto-close}	37
TR12	{open}, {open,respond}, {open,respond,receive}, {open,respond,receive,incident,resolve,auto-close}	1195
TR13	{open}, {open,respond}, {open,respond,receive}, {open,respond,receive,resolve,auto-close}	3169
TR14	{open}, {open,respond}, {open,respond,¬reopen,auto-close}	12
TR15	{open}, {open,respond}, {open,respond,receive}	531
TR16	{open}, {open,respond}, {open,respond,¬instock}	10

For this exercise, the first check is towards the end effect scenario of each trace where in all traces, the end effect must satisfy any one of the goal in the goal model. We can determine from Table 4 that among the 16 distinct traces, the end effect of `TR2`, `TR9` and `TR16` do not conform to any goal. Upon closer inspection, it reveals that some of these traces are not fault or error, but the process is not finished yet and the effects are simply some kind of intermediate state. For example in `TR2` where the end effect is ¬`receive`, the state is to identify that the process is still waiting for user input and has not received any at the observed time.

For the 13 remaining traces, the next check would be whether any one of the effect in the trace conform to a goal. By annotating each effect, we discover that the effect `rejected` of `TR10` does not conform to any goal, therefore we annotate this trace as `exception`, while the 12 other traces are annotated as `normal`.

The last check is to examine whether in the normal trace, the goal precedence constraints in each trace is preserved. We perform the checking between any two consecutive goals (pair-wise) in the trace. From 12 normal traces, we found that all of them are preserving the goal precedence constraints.

Table 3. Goal assertions for the goal model

Goal	Goal assertion
Ticket handled (G0)	close ∨ auto-close
Ticket initiated (G1)	open
Ticket acknowledged and problem assigned (G2)	respond
Requirements provided (G3)	approved ∨ receive ∨ instock
DM approval acquired (G5)	approved
User input acquired (G6)	receive
Stock acquired (G7)	instock
Unresolved problem handled (G9)	auto-close
Problem resolved (G10)	resolve
Request fulfilled (G11)	request ∧ resolve
Incident resolved (G12)	incident ∧ resolve
New ticket created (G13)	reopen
Problem closed (G14)	¬reopen

Table 4. Goal sequence for effect trace

Event sequence name	Goal sequence
TR1	(G1),(G1,G2)
TR2	(G1),(G1,N/A)
TR3	(G1),(G1,G2),(G1,G2,G0)
TR4	(G1),(G1,G2),(G1,G2,G10),(G1,G2,G10,G0)
TR5	(G1),(G1,G10),(G1,G10,G9),(G1,G10,G9,G0)
TR6	(G1),(G1,G5)
TR7	(G1),(G1,G2),(G1,G2,G6,G3),(G1,G2,G6,G3,G14,G9,G0)
TR8	(G1)
TR9	(G1),(G1,N/A)
TR10	(G1),(G1,G2),(G1,G2,N/A),(G1,G2,N/A,G0)
TR11	(G1),(G1,G2),(G1,G2,G13,G9,G0)
TR12	(G1),(G1,G2),(G1,G2,G6,G3),(G1,G2,G6,G3,G12,G0)
TR13	(G1),(G1,G2),(G1,G2,G6,G3),(G1,G2,G6,G3,G11,G0)
TR14	(G1),(G1,G2),(G1,G2,G14,G9,G0)
TR15	(G1),(G1,G2),(G1,G2,G6,G3)
TR16	(G1),(G1,G2),(G1,G2,N/A)

We use these 12 normal traces to build the goal orchestrations model. However, we only use the complete trace, that is all traces that end in the highest goal (G0), therefore we omit TR1, TR6, TR8, and TR15 and left with eight traces to build the orchestrations. We utilize ProM [31] to mine the workflow net.

To determine the consistency between discovered the goal orchestrations with the goal model, we need to establish that all goals in the goal orchestrations

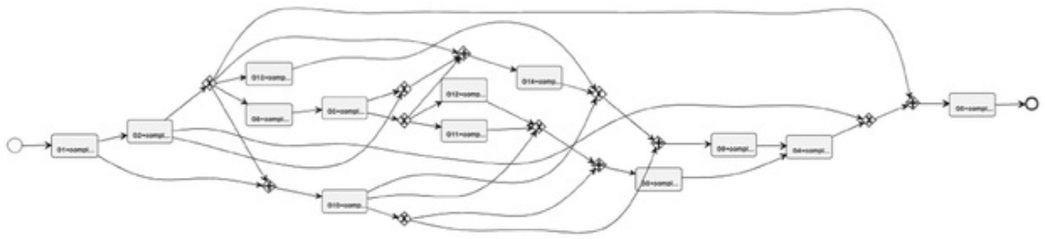

Fig. 3. Goal orchestrations for ticket handling process

presents in the goal model and all transitions preserves the goal precedence constraints in the goal model.

Looking at the goal orchestrations in Fig. 3, there are 13 goals in the goal orchestrations. We confirm that they are also goals in the goal model. The next checking compares the precedence constraints in our library with the transitions in the discovered model. There are 23 transitions between goals in the goal orchestration. Eight of the transitions have a precedence constraint related to them. The checking reveals that these transitions conform to the precedence constraints. The rest of the transitions do not have any constraints related to them. Take for instance, the transitions between `G1` and `G2`. In the goal model, both are sub-goals of `G0`, thus both have precedence over their parent goal, but there is no constraint defined between `G1` and `G2`. Since there is no violation of the goal precedence constraints, we conclude that the discovered goal orchestrations is consistent with the goal model.

6 Related Work

The nearest point of departure for our approach in the literature is the Azzurra framework [5], where business processes are modelled as social interactions between actors. A business process is seen as a coordination of actor's interactions to achieve the established goals. Interactions between actors are governed by commitments. While Azzurra focuses on inter-actor interactions in business processes and the realization of commitments, our work provides in addition a goal-oriented account of business process execution by individual actors.

Our work also builds on prior work on process mining [31] and the discovery of process designs from legacy artefacts [14]. It also builds on prior work on correlating goals with process designs [21].

There is a long history of work on checking goal realization in downstream artefacts, such as tracking goals through their lifecycle [18], management of changes and impact analysis [1], traces in and between requirements models [16,28], demonstrating compliance with some regulations [22], demonstration in real industrial settings [26], etc.

One of the main challenge highlighted by [28] is the need for presenting the traceability information in a clear and concise fashion. In our research, we

represent our traceability problem by leveraging semantic annotation of business process using formal language in CNF.

To asses the goal realization, many frameworks make a comparison between goals with other system artefacts, such as comparison with testing cases [2,4], comparison with design [9], and comparison with code [25,30,34]. In comparison, our research also trace goal realization during or after system run-time, by comparing the task post-conditions defined during design-time with the result of the system execution.

7 Conclusion

In this paper we propose a representation of business process as a coordination of goals called goal orchestrations. This representation gives us a flexibe and context sensitive enactment of processes and convenient for a goal-driven and knowledge-intensive process. We also present a simple method of mining goal orchestrations from event logs. We illustrate this method using a real world setting of a ticket handling system. In our future work, we would like to further explore the mining of goal orchestrations and implement the concept in other application domains, more specifically in clinical setting.

References

1. Allehyani, B., Reiff-Marganiec, S.: Maintaining goals of business processes during runtime reconfigurations. In: Proceedings of the 8th ZEUS Workshop, pp. 21–28 (2016)
2. Arkley, P., Riddle, S.: Tailoring traceability information to business needs. In: Proceedings of the 14th IEEE International Conference Requirements Engineering, pp. 239–244. IEEE (2006)
3. Chopra, S., Ghose, A., Meyer, T.: Non-prioritized ranked belief change. J. Philos. Log. **32**(4), 417–443 (2003)
4. Cleland-Huang, J., Settimi, R., Duan, C., Zou, X.: Utilizing supporting evidence to improve dynamic requirements traceability. In: Proceedings of the 13th IEEE International Conference on Requirements Engineering, pp. 135–144. IEEE (2005)
5. Dalpiaz, F., Cardoso, E., Canobbio, G., Giorgini, P., Mylopoulos, J.: Social specifications of business processes with azzurra. In: 2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS), pp. 7–18, May 2015
6. Dasgupta, A., Ghose, A.K.: Implementing reactive BDI agents with user-given constraints and objectives. Int. J. Agent-Oriented Softw. Eng. **4**(2), 141–154 (2010)
7. Di Francescomarino, C., Ghidini, C., Rospocher, M., Serafini, L., Tonella, P.: Semantically-aided business process modeling. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 114–129. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04930-9_8
8. Di Pietro, I., Pagliarecci, F., Spalazzi, L.: Model checking semantically annotated services. IEEE Trans. Softw. Eng. **38**, 592–608 (2012)
9. Ernst, N.A., Mylopoulos, J., Yu, Y., Nguyen, T.: Supporting requirements model evolution throughout the system life-cycle. In: Proceedings of the 16th IEEE International Requirements Engineering, RE 2008, pp. 321–322. IEEE (2008)

10. Fensel, D., Facca, F.M., Simperl, E., Toma, I.: Web service modeling ontology. Semantic Web Services, pp. 107–129. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19193-0
11. Fensel, D., Lausen, H., Polleres, A., de Bruijn, J., Stollberg, M., Roman, D., Domingue, J.: Enabling Semantic Web Services: The Web Service Modeling Ontology. Springer, Heidelberg (2006). doi:10.1007/978-3-540-34520-6
12. Ghose, A., Goebel, R.: Belief states as default theories: studies in non-prioritized belief change. In: ECAI, vol. 98, pp. 8–12 (1998)
13. Ghose, A., Koliadis, G.: Auditing business process compliance. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 169–180. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74974-5_14
14. Ghose, A., Koliadis, G., Chueng, A.: Rapid business process discovery (R-BPD). In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 391–406. Springer, Heidelberg (2007). doi:10.1007/978-3-540-75563-0_27
15. Ginsberg, M.L., Smith, D.E.: Reasoning about action I: a possible world approach. Artif. Intell. **35**(2), 165–195 (1988)
16. Glorio, O., Pardillo, J., Mazon, J.N., Trujillo, J.: Dawara: an eclipse plugin for using i* on data warehouse requirement analysis. In: Proceedings of the 16th IEEE International Requirements Engineering, RE 2008, pp. 317–318. IEEE (2008)
17. Gou, Y., Ghose, A., Chang, C.-F., Dam, H.K., Miller, A.: Semantic monitoring and compensation in socio-technical processes. In: Indulska, M., Purao, S. (eds.) ER 2014. LNCS, vol. 8823, pp. 117–126. Springer, Cham (2014). doi:10.1007/978-3-319-12256-4_12
18. Hayes, J.H., Dekhtyar, A., Sundaram, S.K., Howard, S.: Helping analysts trace requirements: an objective look. In: Proceedings of the 12th IEEE International Requirements Engineering Conference, pp. 249–259. IEEE (2004)
19. Hepp, M., Leymann, F., Domingue, J., Wahler, A., Fensel, D.: Semantic business process management: a vision towards using semantic web services for business process management. In: IEEE International Conference on e-Business Engineering (ICEBE 2005), pp. 535–540. IEEE (2005)
20. Hinge, K., Ghose, A., Koliadis, G.: Process SEER: a tool for semantic effect annotation of business process models. In: Proceedings of the 13th IEEE International EDOC Conference (EDOC-2009). IEEE Computer Society Process (2009)
21. Koliadis, G., Ghose, A.: Relating business process models to goal-oriented requirements models in KAOS. In: Hoffmann, A., Kang, B., Richards, D., Tsumoto, S. (eds.) PKAW 2006. LNCS, vol. 4303, pp. 25–39. Springer, Heidelberg (2006). doi:10.1007/11961239_3
22. Aoki, T., Traichaiyaporn, K., Chiba, Y., Matsubara, M., Nishi, M., Narisawa, F.: Modeling safety requirements of ISO26262 using goal trees and patterns. In: Artho, C., Ölveczky, P.C. (eds.) FTSCS 2015. CCIS, vol. 596, pp. 206–221. Springer, Cham (2016). doi:10.1007/978-3-319-29510-7_12
23. Meyer, T., Ghose, A., Chopra, S.: Social choice, merging, and elections. In: Benferhat, S., Besnard, P. (eds.) ECSQARU 2001. LNCS, vol. 2143, pp. 466–477. Springer, Heidelberg (2001). doi:10.1007/3-540-44652-4_41
24. Meyer, T., Ghose, A., Chopra, S.: Syntactic representations of semantic merging operations. In: Ishizuka, M., Sattar, A. (eds.) PRICAI 2002. LNCS, vol. 2417, pp. 620–620. Springer, Heidelberg (2002). doi:10.1007/3-540-45683-X_88
25. Mirakhorli, M., Fakhry, A., Grechko, A., Wieloch, M., Cleland-Huang, J.: Archie: a tool for detecting, monitoring, and preserving architecturally significant code. In: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2014, New York, NY, USA, pp. 739–742. ACM (2014)

26. Panis, M.C.: Successful deployment of requirements traceability in a commercial engineering organization... really. In: Proceedings of the 18th IEEE International Requirements Engineering Conference (RE), pp. 303–307. IEEE (2010)
27. Santiputri, M., Ghose, A.K., Dam, H.K.: Mining task post-conditions: automating the acquisition of process semantics. Data Knowl. Eng. **109**, 112–125 (2017)
28. Siegl, S., Hielscher, K.S., German, R.: Model based requirements analysis and testing of automotive systems with timed usage models. In: Proceedings of the 18th IEEE International Requirements Engineering Conference (RE), pp. 345–350. IEEE (2010)
29. Smith, F., Proietti, M.: Rule-based behavioral reasoning on semantic business processes. In: ICAART, SciTePress, pp. 130–143 (2013)
30. Valderas, P., Pelecha, V., Pastor, O., et al.: Requirements engineering for pervasive systems. A transformational approach. In: Null, pp. 351–352. IEEE (2006)
31. Van der Aalst, W., Weijters, T., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE Trans. Knowl. Data Eng. **16**(9), 1128–1142 (2004)
32. Weber, I., Hoffmann, J., Mendling, J.: Beyond soundness: on the verification of semantic business process models. Distrib. Parallel Databases **27**, 271–343 (2010)
33. Winslett, M.: Reasoning about action using a possible models approach. Urbana **51**, 61801 (1988)
34. Yu, Y., Wang, Y., Mylopoulos, J., Liaskos, S., Lapouchnian, A., do Prado Leite, J.C.S.: Reverse engineering goal models from legacy code. In: Proceedings of the 13th IEEE International Conference on Requirements Engineering, pp. 363–372. IEEE (2005)

Goal Orchestrations: Modelling and Mining Flexible Business Processes

ORIGINALITY REPORT

12%

SIMILARITY INDEX

PRIMARY SOURCES

1	"Conceptual Modeling", Springer Nature America, Inc, 2015 Crossref	191 words — 3%
2	"Business Process Management", Springer Nature America, Inc, 2015 Crossref	50 words — 1%
3	Yingzhi Gou, Aditya Ghose, Hoa Khanh Dam. "Chapter 29 Leveraging Game-Tree Search for Robust Process Enactment", Springer Nature, 2017 Crossref	40 words — 1%
4	João Araújo. "NORMAL SEMIGROUPS OF ENDOMORPHISMS OF PROPER INDEPENDENCE ALGEBRAS ARE IDEMPOTENT GENERATED", Proceedings of the Edinburgh Mathematical Society, 02/2002 Crossref	28 words — < 1%
5	Metta Santiputri, Novarun Deb, Muhammad Asjad Khan, Aditya Ghose, Hoa Dam, Nabendu Chaki. "Chapter 6 Mining Goal Refinement Patterns: Distilling Know-How from Data", Springer Nature, 2017 Crossref	28 words — < 1%
6	"Conceptual Modeling", Springer Nature, 2016 Crossref	28 words — < 1%
7	"Transactions on Petri Nets and Other Models of Concurrency XI", Springer Nature, 2016 Crossref	24 words — < 1%
8	Wil M. P. van der Aalst. "Conformance checking of service behavior", ACM Transactions on Internet Technology, 05/01/2008 Crossref	23 words — < 1%
9	Lecture Notes in Computer Science, 2011. Crossref	16 words — < 1%
10	"Enterprise, Business-Process and Information Systems Modeling", Springer Nature America, Inc, 2014 Crossref	16 words — < 1%

11	Lecture Notes in Computer Science, 2012. Crossref	15 words — < 1%
12	Lecture Notes in Business Information Processing, 2016. Crossref	15 words — < 1%
13	Lecture Notes in Computer Science, 2008. Crossref	14 words — < 1%
14	Lecture Notes in Business Information Processing, 2014. Crossref	14 words — < 1%
15	Sunil Nair, Jose Luis de la Vara, Sagar Sen. "A review of traceability research at the requirements engineering conference^{re@21}", 2013 21st IEEE International Requirements Engineering Conference (RE), 2013 Crossref	13 words — < 1%
16	"Advanced Information Systems Engineering", Springer Nature, 2017 Crossref	11 words — < 1%
17	Alves de Medeiros, A.K.. "Quantifying process equivalence based on observed behavior", Data & Knowledge Engineering, 200801 Crossref	10 words — < 1%
18	www.heppnetz.de Internet	10 words — < 1%
19	Wassim Derguech. "An Indexing Structure for Maintaining Configurable Process Models", Lecture Notes in Business Information Processing, 2010 Crossref	9 words — < 1%
20	Lecture Notes in Electrical Engineering, 2016. Crossref	9 words — < 1%
21	etheses.whiterose.ac.uk Internet	9 words — < 1%
22	Xuewen Xia, Yuanxiang Li, Jixiang Zhu. "A high-quality pseudorandom numbers generator based on twi-layer couple cellular automata", 2009 IEEE Congress on Evolutionary Computation, 2009 Crossref	9 words — < 1%
23	Web Services Foundations, 2014. Crossref	9 words — < 1%
24	dkm-static.fbk.eu	

	Internet	9 words — < 1 %
25	pdfs.semanticscholar.org Internet	9 words — < 1%
26	Gang Chen, Guoqiang Bai, Hongyi Chen. "A dual-field elliptic curve cryptographic processor based on a systolic arithmetic unit", 2008 IEEE International Symposium on Circuits and Systems, 2008 Crossref	8 words — < 1%
27	"Advances in Artificial Intelligence: From Theory to Practice", Springer Nature, 2017 Crossref	8 words — < 1%
28	Lecture Notes in Business Information Processing, 2013. Crossref	8 words — < 1%
29	Lecture Notes in Computer Science, 2014. Crossref	8 words — < 1%
30	Lecture Notes in Computer Science, 2010. Crossref	8 words — < 1%
31	www.uow.edu.au Internet	8 words — < 1%
32	Chen, Liping, Guojun Zhang, and Weitao Ha. "Conformance checking for interaction of web service composition with temporal logic", International Journal of Sensor Networks, 2014. Crossref	7 words — < 1%
33	"Information Systems", Springer Nature, 2017 Crossref	6 words — < 1%
34	Lanoue, Daniel Parmet. "The Metric Coalescent.", Proquest, 2015. ProQuest	6 words — < 1%
35	Lecture Notes in Business Information Processing, 2012. Crossref	6 words — < 1%

EXCLUDE QUOTES OFF
EXCLUDE MATCHES OFF
EXCLUDE BIBLIOGRAPHY ON