Refinement Mining: Using Data to Sift Plausible Models

Antonio Cerone(✉)

Department of Computer Science, Nazarbayev University, Astana, Kazakhstan
antonio.cerone@nu.edu.kz

Abstract. Process mining techniques have been developed in the ambit of business process management to extract information from event logs consisting of activities and then produce a graphical representation of the process control flow, detect relations between components involved in the process and infer data dependencies between process activities. These process characterisations allow the analyst to *discover* an annotated visual representation of the conceptual model or the performance model of the process, *check conformance* with an *a priori* model to detect deviations and *extend* the *a priori* model with quantitative information such as frequencies and performance data. However, a process model yielded by process mining techniques is more similar to a representation of the process behaviour rather than an actual model of the process: it often consists of a huge number of states and interconnections between them, thus resulting in a spaghettilike net which is hard to interpret or even read.

In this paper we propose a novel technique, which we call *model mining*, to derive an abstract but concise and functionally structured model from event logs. Such a model is not a representation of the unfolded behaviour, but comprises, instead, a set of formal rules for generating the system behaviour. The set of rules is inferred by sifting a plausible *a priori* model using the event logs as a sieve until a reasonably concise model is achieved (*refinement mining*). We use rewriting logic as the formal framework in which to perform model mining and implement our framework using the MAUDE rewrite system. Once the final formal model is attained, it can be used, within the same rewriting logic framework, to predict future evolutions of the behaviour through simulation, to carry out further validation or to analyse properties through model checking. We illustrate our approach on a case study from the field of ecology.

Keywords: Formal methods · Model-driven approaches · Process mining · Application to ecosystem modelling

1 Introduction

The large amount of data available in online repositories have recently driven research towards the development of techniques and methodologies aiming at

P. Milazzo et al. (Eds.): STAF 2016, LNCS 9946, pp. 26–41, 2016.
DOI: 10.1007/978-3-319-50230-4_3

extracting meaningful information from data and exploiting it to describe and understand the processes that have generated such data. Large online repositories exist in various areas, ranging from economy, learning, sociology and other social sciences to biology, medicine and ecology.

One of these data analysis techniques is *process mining*, which emerged in the field of business process management (BPM). It is used to extract information from event logs consisting of activities and then produce a graphical representation of the process control flow, detect relations between components/individuals involved in the process and infer data dependencies between process activities [17]. Process mining supports not only the *discovery* of an *a posteriori* process model and its representation as a process map, but also the *extension* of a pre-existing *a priori* model by enriching it with new aspects and perspectives and the comparison, by using a technique called *conformance analysis* [13], of the *a priori* model with the event logs.

These three approaches, i.e. discovery, extension and conformance analysis, are alternative ways of using process mining and, although it is possible to apply them all to the same case, their outcomes cannot be automatically integrated, but require the analyst to proceed manually with a comparison work [10].

Only recently the potentialities of process mining in disciplines other than BPM are starting to be understood and realised. Process mining and conformance analysis can be used in a number of contexts, in and across areas such as human-computer interaction (HCI) and learning [3]. Some of these ideas have already resulted in research outputs: process mining have been used to extract learning processes from open source software (OSS) project repositories and check conformance of learning models based on the literature [10,11].

Process mining aims at understanding the process and validating its efficiency. It is therefore appropriate for *descriptive* purposes but not for actual modelling. We want to take a step forward and exploit real data in a *constructive* rather than descriptive way by integrating techniques from the realm of process mining with modelling approaches. In particular, we are interested in formal approaches to modelling, which have the potential to make the modelling process automatic, by appropriately manipulating the information extracted from data logs, and produce a model that can be automatically verified. Although there are a number of works in the areas of synthesis of programs [7,14,15] and synthesis of biological and probabilistic systems from data [5,8,12], to our knowledge, the only attempt to integrate process mininig and formal verification is a work by van der Aalst, de Beer and van Dongen's, which aims at verifying whether an event log satisfies a property expressed in linear temporal logic (LTL) [16]. To accomplish our objective we use rewriting logic [9], and the Maude system [6], a tool based on rewriting logic that has a great expressive power and is equipped with an efficient model checker.

Section 2 introduces the notion of structured event used in our framework and shows how to instantiate it within various application fields, also providing motivations for our work. Section 3 introduces our Model Mining Formal Framework (MMFF) by defining the required data structure while Sect. 3.1 describes

the model mining engine. Section 4 presents how to sift a plausible *a priori* model using event logs and illustrates the approach using a case study on the dynamics of a mosquito population. Section 5 describes the possible uses of the generated model and the advantages of the model mining approach to modelling. Finally, Sect. 6 illustrate further possible applications and future research challenges.

2 Event Structure and Instantiation

In order to be successfully processed with the purpose of extracting process control flow information, event logs have to meet a number of structural properties, namely to contain adequately organised and clustered data. Therefore, unstructured data contained in event logs stored in repositories have first to be semantically interpreted according to the purpose of the model we aim to devise, so that such interpretation can drive their structuring and clustering. Structural and semantical organisation can be attained by applying text mining techniques, in particular semantic indexing, in combination with an appropriate ontology from the given application domain. This approach is commonly used in process mining, which is thus applied to a set of *pre-processed* event logs.

For the purpose of model mining we assume to have already pre-processed events organised as a sequence of structured entities. In order to devise a general methodology to efficiently apply formal methods to drive the mining engine, we consider an essential structure of events, as a list of attributes that are general enough to be instantiated depending on the application domain and purpose. In particular, we aim to define a methodology that is partly independent of the precise semantics of the attribute.

We consider the reality to be modelled as an ecosystem of dynamic entities linked together through causality relationships and activity flows. These causality relationships and activity flows are what we need to discover in order to build the ecosystem model. The dynamic evolution of the ecosystem, namely of its entities, is visible through events.

An *event* is defined as a quadruple

$$[|t, d, s, v|]$$

where

- t is the *time* at which the event occurs;
- d is the *domain name*, the name of the domain to which entities belong;
- s is the *subdomain name*, namely a further categorisation of the entity within the domain;
- v is the *concrete value* that refers to domain and subdomain.

We consider two level of domains (domain and subdomain), but this could be generalised to n levels.

Moreover, we have entities which are the *target* of our modelling process and others that make up the *environment* in which our target entities evolve. Thus we identify two categories of events:

target event which describes the activities or actions to be modelled;
environmental event which modifies the conditions that enable activities and actions.

We denote an environmental event by $env[|t, d, s, v|]$ and a target event by $target[|t, d, s, v|]$. We describe below possible instantiations of target and environmental events in various application fields, as summerised in Table 1, in which a "Value" within the "Domain" quantifies or qualify the event, while "Weight" quantifies a relevant attribute of the "Value" within the environment.

In the field of ecology, if we wish to model the dynamic of a population, target events may describe the population size (hence its growth activity) or location (hence its movement or migration activity). In a typical target event, the domain is the population species, the subdomain is the maturation stage of the individual (e.g. larva, juvenile or adult) and the concrete value is the actual population size. Environmental events determine or modify all conditions that affect the life and maturation of the population (e.g. food availability, predation, human impact, temperature, humidity, etc.).

For more examples, we can consider the realm of social networks. In a review community (about resorts, hotels, restaurants, books, or specific products) activities are reviews, ratings, replies, recommendations, while possible environmental conditions are changes to or launch, closing/discontinue of services/products. In an online product support forum, activities are postings and replies while possible environmental conditions are release of upgrades and new products.

One of the most popular social networking website is Facebook. If we wish to analyse the friendship relationships (target domain) between community members, we need to characterise the level of friendship or the trustiness or reputation of the friend (value). Conditions that affect these values are intensity, richness/rate and valence (weight) of posts, "likes" and comments (value) on the friend's wall (environmental domain). Intensity comprises the frequency and size of the post/like/comment, richness refers to the contents (e.g. text, stickers, photos) and valence is the emotional impact (positive, neutral or negative).

A very articulated case of social network is an OSS community. Project contributors perform a large range of activities: post, reply to a post, send a message, review code and commit code. If we consider these activities as environmental domains, as in Table 1, with aspects of their contents as value, and their persistence (posts, replies and messages persist when they generate conversations/interactions) and impact (the impact of reviews and commits is their effectiveness in improving the code) as weights, these can be seen as conditions that affect the growth of individual community members in terms of knowledge, expertise and skills, the evolution of the entire community, as well as the project productivity and, to a greater extent, the quality of the produced software [2]. Table 1 shows how these environmental conditions affect domains in the ambit of collaborative learning: learning process and skill acquisition. The learning process can be described in terms of learning stages. Examples of learning stages of a project contributor are: "understanding", the initial stage wherein the contributor observe community activities, use the code, exchange emails and

post messages with the purpose of understanding contents but does not produce new content; "practising", wherein the contributor proposes new contents and production activity starts as a trial and error process; and "developing", an advanced stage wherein the contributor already commits code [2]. Skill acquisition can be described in terms of the activities carried out by the contributor: coding, reviewing, etc.

Table 1. Example of event instantiations depending on the environment and target domains from a number of application fields.

Applic. Field	Environmental			Target		
	Domain	Value	Weight	Domain	Value	Activity
Ecology	Food	availability	duration	Population	location	migrate
	Temper.	level	duration		size	grow
	Predation	life impact	periodicity			*intervent. measures*
	Human Impact	habitat deterior.	extension			
			severity			
Emerg. Manag.	Quake	intensity	frequency	Casualties	number	monitor
	Rain	amount	water persistence	Damage	amount	
		intensity	duration		severity	*response activities*
	Temper.	level	duration	Bushfires	extension	
Social Network (Facebook)	Wall	post	intensity	Friendship	level	increase or decrease
		like	richness/rate		trust	
		comment	valence		reputation	
Collab. Learning in OSS Commun.	Post	content aspects	persistence (conversation, interaction)	Learning Process	learning stage	achive maturation
	Reply					
	Message				or	
	Review		impact (effectiveness)	Skill Acquisition	contributor activity	acquire skill
	Commit					
HCI	Task	outcome	frequency	Interface	state	transition
Cognitive Science	Interface	experience	frequency	Mental Model	interface state	human action

In the field of HCI, state and transition of an interface depend on the tasks performed by the human in interacting with the interface, with the value representing the outcome of the task (successful outcome or failures) and a possible weight given by the frequency with which the task is performed. In the more general view of human behaviour considered in cognitive science, humans have the skill of exploiting their experiences to build mental models of the reality. In a way, we can say that humans realise a form of "model mining" when building models out of experiential data. Interacting with a specific interface produces this kind of experiencial data and allows the human to build a mental model of the way the interface works. We can describe this situation in our framework as shown in the last row of Table 1.

The last column of Table 1 shows the activities that refer to the target domains. Most of these activities are *descriptive*, namely they describe how the domain values change. For example, if we consider a population as a domain, a change in its location is a migration, while a change in its size is a growth. Activities shown in italic are instead *responsive*, namely they aim to modify or, at least, restrain the descriptive activity. For example, in ecology, intervention measures are carried out to favour or control the growth of a population.

3 Model Mining Formal Framework

In order to characterise the effect of events on their domain, we need to define the notion of domain state. Moreover, while events are concrete entities that accurately represent the reality, states rather refer to the model than to the reality. They are thus abstract entities, whose values are abstract values.

A *domain state* is defined as a quintuple

$$(|d, s, a, w, f|)$$

where

- d is the *domain name*;
- s is the *subdomain name*;
- a is the *abstract value* that refers to domain and subdomain;
- w is the *weight* of the value.
- f, called *weighting function*, is a function from abstract domains to weights.

We mentioned in Sect. 2 that the *weight* quantifies a relevant attribute of the value within the environment. Thus the *weight* quantifies a relevant attribute of the *abstract value*. Weight is initialised by f at each change of abstract value and normally varies as time progresses, independently of the occurrence of events. For example, in Table 1, we note that in Ecology, a relevant attribute of the value for "Food availability" is "duration", since it is the duration of food availability that modifies the state of the environment. Similarly, relevant attributes of the value for "Habitat deterioration" are "extension" and "severity" and, in the area of Collaborative Learning in OSS Communities, relevant attributes of a "Post content" are its "persistence" in conversations and interactions and its "impact" on the OSS product, in terms of effective contribution to its improvement.

We denote an *environmental state* by $env(|d, s, a, w|)$ and a *target state* by $target(|d, s, a, w, f|)$. The *global state* of the ecosystem we are modelling consists of a set of domain states.

Concrete and abstract values are linked by an *abstraction relation*

$$\{d, s \mid v_1, v_2 \rightarrow a\},$$

where v_1 and v_2 may be distinct only if domain d is totally ordered and in this case $v_1 \leq v_2$. This relation maps any value v of domain d and subdomain s such that $v_1 \leq v \leq v_2$ to abstract value a.

The occurrence of environmental event $env[|t, d, s, v|]$ changes environmental state $env(|d, s, a_0, w_0|)$ to $env(|d, s, a, w|)$ iff there exist an abstraction relation

$$\{d, s \mid v_1, v_2 \rightarrow a\} \text{ such that } v_1 \leq v \leq v_2,$$

such that

$$w = \begin{cases} f(a) \text{ if } a \neq a_0 \\ w_0 \quad \text{otherwise} \end{cases} \tag{1}$$

Therefore, the state weight is reset to $f(a)$ only if there is a change in the abstract state. In the examples in this paper, we will always have $f \equiv 0$ and represent state $(|d, s, a, w, f|)$ simply as a quadruple $(|d, s, a, w|)$. The state weight is regularly incremented by the passage of time.

We use rewriting logic to define the state transitions. Since one purpose of model mining is the application of model refinement to a plausible set of models, we need to consider sets of alternative plausible rules.

A *plausible rule* is defined as

$$\{i \mid pre_1, ..., pre_n \Rightarrow post\}$$

where

- i is the *identification number* of the plausible rule;
- $pre_i = [d_i, s_i, a_i, \tau_i]\rhd$ are *preconditions*, with d_i and s_i environmental domain and subdmain, respectively, for $i = 1, \ldots, n$, and τ_i denoting a threshold $\mid \tau_i \mid$ for the effect of the precondition abstract value on the postcondition;
- $post = \ll d, s \mid a_0 \xrightarrow{\alpha} a \gg$ is a *postcondition*, with d and s target domain and subdomain, respectively.

The plausible rule is enabled on environmental state $(|d_0, s_0, a_0, w_0|)$ iff, for each precondition

$$pre_i = [d_i, s_i, a_i, \tau_i]\rhd, \ i = 1, \ldots, n,$$

such that $d_i = d_0$, $s_i = s_0$ and $a_i = a_0$, the following conditions hold

1. $\tau_i - w_0 \geq 0$ if $\tau_i > 0$;
2. $\tau_i + w_0 \geq 0$ if $\tau_i \leq 0$.

The application of the rule causes a transition from state $(|d_0, s_0, a_0, w_0|)$ to state $(|d_0, s_0, a, w|)$, where w is defined as in Eq. (1) above.

A positive threshold $\tau_i > 0$ means that the precondition is satisfied only if the abstract value a_0 in the current state has been persisting for at most a time w_0 (Condition 1 above is satisfied). A non positive threshold $\tau_i \leq 0$ means that the precondition is satisfied only if the abstract value a_0 in the current state has been persisting for at least a time w_0 (Condition 2 above is satisfied).

In order to illustrate the definitions presented in this section, let us consider the following example. Let $env(|Temp, avg, med, 3|)$ be the temperature state at day 131. It denotes that astract value *med*, defined according to the

abstraction relation given in Table 2, persisted for 3 days. Environmental event $env([|131, Temp, avg, 25|])$, which describes a concrete temperature value of 25° at day 131, changes such temperature state to $env(|Temp, avg, high, 0|)$.

Let us consider an application to ecology where the target domain is represented by a mosquito population of species *Aedes albopictus*, which rapidly increases in size when the temperature is high. Let us consider the abstraction relations in Table 2. Plausible rule

$$\{1 \mid [Temp, avg, high, -10]\triangleright \Rightarrow\ll Aedes, adult \mid med \overset{increase}{\longrightarrow} high \gg\}$$

has a single precondition and states that if a high temperature persists for at least 10 days, then the size of the adult population of *Aedes a.* increases from medium to high. If between day 131 and day 141 there are no events that change the temperature state, $env(|Temp, avg, high, 0|)$ at day 131 becomes $env(|Temp, avg, high, 10|)$ at day 141. The latter state satisfies Condition 2 above, thus, at day 141, a population state $env(|Aedes, adult, med, w|)$ would be changed by the plausible rule above to $env(|Aedes, adult, high, 0|)$.

3.1 The Model Mining Engine

We perform model mining using rewriting logic. A prototype of the model mining engine, implemented using the MAUDE rewrite system [6], and its application to the case study presented in Sect. 4.1 can be downloaded at

http://sysma.imtlucca.it/refinement-mining-datamod-2016/.

We consider discrete time with a granularity suitable to the specific application domain. For example, a daily time progress would, in many cases, suit the analysis of the dynamic of a population, while for a social network the granularity depends on the frequency of the activity relevant to the model we want to capture. The data structures defined in Sect. 3 are organised in a *configuration*, which is manipulated by the model mining engine using rewrite rules. To distinguish such rules from the plausible rules we call them *meta-rules*.

A configuration comprises:

abstraction a set of abstraction relations;
past events a list of events that have already been processed by the current step of the engine;
current events a list of events that are currently being processed by the current step of the engine;
future events a list of events that have not been processed yet by the current step of the engine;
option set a collection of plausible rules stuctured as described in Sect. 4;
current state the global state at the current time, which consists in a set of domain states;

control containing *static information* for reset purpose, i.e. initial domain state for the target domain, initial time, start and end time for simulation and model mining, and *dynamic information*, i.e. information to control the meta-rule selection and the current choice of plausible rules as well as incrementally built **constructed model** and **refinement**, and possibly a list of **rejected models**.

The engine makes use of the following meta-rules:

1. **Time Progress** to increase the *current time* and the weight of every domain state at the beginning of each step and move the events occurring at the next time from **future events** to **current events**;
2. **Init Simulation** to initialise **future events** as the list of all events and **control** with the initial and final times for the simulation;
3. **State Update via Events** to modify the current state using environmental events and abstraction relations as shown in Sect. 3, if there are any environmental events in **current events**;
4. **State Update via Model** to select a plausible rule with preconditions satisfying the **current state** and modify the current state as shown in Sect. 3;
5. **No State Update via Model** to skip the selection of a plausible rule with preconditions not satisfying the **current state**;
6. **Model Validation via Events** to compare the current state with the target events, if there are any target events in **current events**;
7. **Init Model Refinement** initialise **future events** as the list of all events and **control** with the initial and final times for the model refinement;
8. **Refinement Step** to sift the plausible model using the target events, if there are any target events in **current events**.

Meta-rules 2–6 perform the simulation and meta-rules 7 and 8 perform the model refinement. The architecture of our model mining engine is shown in Fig. 1. Each component consist of the evaluation and possibly application of the meta-rule with the same name, apart form component **State Update via Model** which comprises meta-rules 4 and 5.

4 Refinement Mining

Refinement mining is a data-driven model refinement that consists in sifting a plausible *a priori* model using the event logs as a sieve until a reasonably concise model is achieved. In order to carry out refinement mining, plausible rules must be structured into sets of alternatives called *option sets*. An *option set* is defined as

$$[i]\langle pRule_1, ..., pRule_n\rangle$$

where

– i is the *identification number* of the option set;
– $pRule_i$ are alternative plausible rules for $i = 1, \ldots, n$.

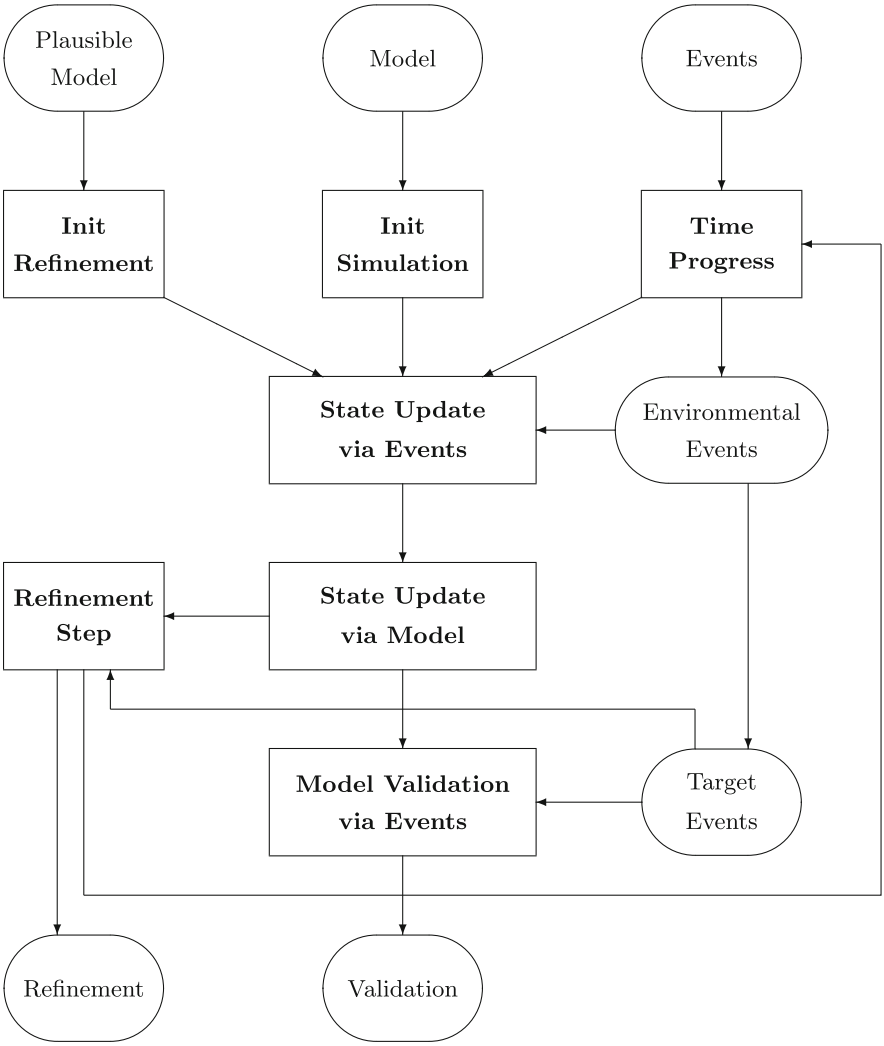

Fig. 1. Architecture of the model mining engine.

A *plausible model* is a set of option sets.
A *rule reference* is defined as

$$[n : i(j)]$$

where

– n is a reference to the option set whose identification number is n;
– i is a reference to the plausible rule whose identification number is i within the option set identified by n;

– j is the number of times the referred plausible rule has been applied during the current simulation.

A *model reference* is a set of rule references, one for each option set.
A *refinement* is a set of model references.

Refinement mining is carried out by performing an event-driven simulation on all possible choices of one plausible rule for each option set. Each choice is represented by a *model reference*. At each step of the simulation, if there are target events in the **current state**, then component **Refinement Step** of the model mining engine compares the concrete value in each of such target events with the simulated abstract value of the corresponding target state: if the concrete value is covered by the abstract value then the current *model reference* is added to **refinement**, otherwise it is added to **rejected models** (**refinement** and **rejected models** are part of the **control** field in the engine configuration). Refinement mining is completed once all possible choices of models are simulated.

4.1 A Case Study from Ecology

In previous work [1] we modelled the dynamic of a population of *Aedes albopictus*, a mosquito species known as "tiger mosquito", which is endemic of Asian regions, where it is a carrier of dengue fever, and now widespread also in Europe. The model developed in that work is based on biological aspects of the mosquito and considers the impact of changes in the environmental conditions on such biological aspects to simulate the population dynamics. Among relevant environmental conditions are average temperature and rain amount. The simulation made use of data on the size of the mosquito population collected during May–November 2009 in the province of Massa-Carrara (Tuscany, Italy) using CO2 mosquito traps.

If we consider average temperature and rain as environmental domains and the mosquito population as target domain, we can envisage a plausible model whose option sets characterise the possible effects of average temperature and rain amount on the mosquito population. A possible option set is as follows.

$$
\begin{aligned}
[n]\ \langle\ & \{1 \mid [Temp, avg, high, -10] \triangleright \Rightarrow\!\ll Aedes, adult \mid high \overset{increase}{\longrightarrow} extr \gg\} \\
& \{2 \mid [Rain, amount, high, +5] \triangleright \Rightarrow\!\ll Aedes, adult \mid high \overset{increase}{\longrightarrow} extr \gg\} \\
& \{3 \mid [Temp, avg, high, -1] \triangleright, [Rain, amount, high, +10] \triangleright \\
& \qquad \Rightarrow\!\ll Aedes, adult \mid high \overset{increase}{\longrightarrow} extr \gg\} \\
\rangle\ &
\end{aligned}
$$

The three alternative plausible rules of option set n state that

1. if a high temperature persists for at least 10 days, then the size of the adult population of *Aedes a.* increases from high to extreme;
2. if a high rainfall occurred at most 5 days earlier, then the size of the adult population of *Aedes a.* increases from high to extreme;

3. if a high temperature persists for at least 1 day, and a high rainfall occurred at most 10 days before, then the size of the adult population of *Aedes a.* increases from high to extreme;

If we consider the sequence of events

$$env[|56, Temp, avg, 25|] \longrightarrow target[|56, Aedes, adult, 360|] \longrightarrow$$
$$env[|59, Temp, avg, 24|] \longrightarrow env[|59, Rain, amount, 72|] \longrightarrow$$
$$env[|60, Temp, avg, 23|] \longrightarrow env[|61, Temp, avg, 22|] \longrightarrow$$
$$env[|62, Temp, avg, 23|] \longrightarrow env[|64, Temp, avg, 24|] \longrightarrow$$
$$env[|66, Temp, avg, 25|] \longrightarrow env[|67, Temp, avg, 26|] \longrightarrow$$
$$target[|67, Aedes, adult, 561|]$$

which is a fragment of the data collected in 2009 in the province of Massa-Carrara and the abstraction relations defined in Table 2, we obtain the abstract values and weights shown in Table 3. We note that only plausible rule 3 is applicable at day 67 and thus validated by the sequence of events (i.e. by the data). In fact, plausible rule 1 is not applicable because the high temperature persisted for just 1 day rather than the minimum of 10 required by the rule precondition; plausible rule 2 is not applicable because the high rainfall occurred already 7 days earlier rather than the maximum of 5 required by the rule precondition. If we suppose that the plausible model consists of just the option set n above and the refinement mining is driven only by the dataset given in Table 3, then the final refinement would be set $\{[n : 3(1)]\}$, which consists of just the rule reference of 3 with 1 as the number of times rule 3 has been applied during the only possible simulation.

5 Model Usage and Model Mining Advantages

The purpose of our framework is not only to generate a model refinement but also to use it in three possible ways.

The most obvious usage is for *prediction*. Simulation can be run to predict the behaviour of the target domain.

Table 2. Abstraction relations for average temperature (*Temp*, *avg*), amount of daily rain (*Rain*, *amount*) and *Aedes a.* population size (*Aedes*, *adult*).

Abstract value	Concrete value for		
	Temp, *avg*	*Rain*, *amount*	*Aedes*, *adult*
low (low)	0–19	0–5	0–100
med (medium)	20–24	6–40	101–300
high (high)	25–35	41–200	301–500
extr (extreme)	36–50	201–500	501–800

Table 3. Abstract values (abs) and weight (weight) for real data average temperature (*Temp*, *avg*), amount of daily rain (*Rain*, *amount*) and *Aedes a.* population size (*Aedes*, *adult*), collected in 2009 in the province of Massa-Carrara (the abstract values are added in accordance with the abstraction relations in Table 2).

Date	Day no.	*Temp*, *avg*			*Rain*, *amount*			*Aedes*, *adult*	
		val	abs	weight	val	abs	weight	val	abs
3 July	56	25	high	?	-	-	0	360	high
6 July	59	24	high	> 0	72	high	1	-	-
7 July	60	23	med	0	-	-	2	-	-
8 July	61	22	med	1	-	-	3	-	-
9 July	62	23	med	2	-	-	4	-	-
11 July	64	24	med	3	-	-	5	-	-
13 July	66	25	high	0	-	-	6	-	-
14 July	67	26	high	1	-	-	7	561	extr

Another important usage is *model validation*. Although the refinement process is based on validation against real data (target events), the resultant model may need further validation due to changes in environmental domains as well as in the target domain. This is a common situation both in biological/ecological contexts and in socio-economic contexts. In ecological contexts the environment changes due to natural degradation and human interventions while populations get adapted to new environments. In socio-economic contexts the continuous development of new technologies changes the environmental conditions while the users of such technologies, on the one hand, get adapted to them and, on the other hand, invent new ways to use them, often ways that the designer themselves could not predict. In these continuously evolving contexts, an important advantage of model mining is that the generated model can be revised through further validation and, if the model is invalidated by the data, then model mining may be run again on the initial plausible model or on a revision of it.

The last possible usage of the generated model refinement is *property checking*. This is possible by exploiting the model checking capabilities of the MAUDE rewrite system by which plausible models are manipulated.

6 Conclusion and Future Work

We defined a formal framework (MMFF), based on rewriting logic and implemented in the MAUDE rewrite system, for generating a data-validated model starting from a plausible model defined *a priori*. We illustrated MMFF on a case study from the field of ecology.

As our future work, we are planning to apply MMFF to the various fields mentioned in Sect. 2 and illustrated in Table 1, in particular to collaborative learning, HCI and cognitive science. These applications will build on our previous work and aim to investigate how refinement mining would scale to large datasets. The initial plausible model may either be based on theoretical hypotheses or be manually built through the observation of the dataset. In the former case refinement mining can be used to verify such hypotheses. In the latter case, however, the larger the dataset is the harder the definition of the plausible model is, which possibly makes it difficult to apply refinement mining to big data.

In our previous work on the application of rewriting logic (also using the MAUDE system) to HCI and cognitive science [4] we have characterised the behaviour of a user that exploits a pre-defined mental model and compared two interface designs with respect to the cognitive errors that may emerge during interaction. We plan to use model mining to extend such work, on the one hand by exploiting environmental events generated by the outcome of human tasks carried out on a set of different interfaces and target events generated by the interface states to identify the interfaces that support the successful completion of the task and, on the other end, by extracting the mental model from event logs produced by interactions between user and interface.

In our previous work on collaborative learning in OSS communities [11] we analysed participants interaction and knowledge exchange in emails repositories of OSS projects by retrieving data carrying information on the learning activities that occur in distinct phases of the learning process to produce pre-processed event logs. Such event logs were fed to a process mining tool to produce visual workflow nets that represent the traces of learning activities in OSS as well as their relevant flow of occurrence. However, such workflow nets consist of huge numbers of states and interconnections between them, which make them hard to interpret. To overcome this problem we plan to revisit our process mining work and transfer our mining approach to MMFF.

Finally, we must note that there are situations in which target domains do not directly correspond to observable events. This happens, for example, in social contexts, when social relationships either are characterised from an introspective point of view (e.g. friendship), which is not directly externalised through events (e.g. friendship level, trust and reputation are not directly observable in events), or evolve over a long time (e.g. learning process), with no events characterising the change of values (learning stages are not directly observable in events). With reference to Table 1 we can observe neither a target event of domain Friendship, whose value directly describes a change in level of friendship or trust, nor a target event of domain Learning Process, whose value directly describes a change of learning stage. Since in these situations there are no observable target events, refinement mining cannot be used.

Therefore, in our future work, we also plan to investigate the possibility of *directly constructing* the model from the event logs, rather than extracting it from a plausible model through refinement mining. We hope that this would

allow model mining to both scale well with big data and deal with target domains that do not feature observable events.

References

1. Basuki, T.A., Cerone, A., Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Rossi, E.: Modelling the dynamics of an Aedes albopictus population. In: Proceedings of AMCA-POP 2010, Electronic Proceedings in Theoretical Computer Science, vol. 227, pp. 37–58 (2010)
2. Cerone, A.: Learning and activity patterns in OSS communities and their impact on software quality. In: Proceedings of OpenCert 2011, ECEASST, vol. 48 (2012)
3. Cerone, A.: Process mining as a modelling tool: beyond the domain of business process management. In: Bianculli, D., Calinescu, R., Rumpe, B. (eds.) SEFM 2015. LNCS, vol. 9509, pp. 139–144. Springer, Heidelberg (2015). doi:10.1007/978-3-662-49224-6_12
4. Cerone, A.: A cognitive framework based on rewriting logic for the analysis of interactive systems. In: De Nicola, R., Kühn, E. (eds.) SEFM 2016. LNCS, vol. 9763, pp. 287–303. Springer, Heidelberg (2016). doi:10.1007/978-3-319-41591-8_20
5. Češka, M., Dannenberg, F., Kwiatkowska, M., Paoletti, N.: Precise parameter synthesis for stochastic biochemical systems. In: Mendes, P., Dada, J.O., Smallbone, K. (eds.) CMSB 2014. LNCS, vol. 8859, pp. 86–98. Springer, Heidelberg (2014). doi:10.1007/978-3-319-12982-2_7
6. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: The Maude 2.0 system. In: Nieuwenhuis, R. (ed.) RTA 2003. LNCS, vol. 2706, pp. 76–87. Springer, Heidelberg (2003). doi:10.1007/3-540-44881-0_7
7. Gulwani, S.: Automating string processing in spreadsheets using input-output examples. In: Notices, A.S. (ed.) Proceedings of POPL 2011, vol. 46, pp. 317–330. ACM (2011)
8. Koksal, A.S., Pu, Y., Srivastava, S., Bodik, R., Fisher, J., Piterman, N.: Automating string processing in spreadsheets using input-output examples. In: Notices, A.S. (ed.) Proceedings of POPL 2013, vol. 48, pp. 469–482. ACM (2013)
9. Martí-Oliet, N., Meseguer, J.: Rewriting logic: roadmap and bibliography. Theor. Comput. Sci. **285**(2), 121–154 (2002)
10. Mukala, P.: Process models for learning patterns in FLOSS repositories. Ph.D. thesis, Department of Computer Science. University of Pisa (2015)
11. Mukala, P., Cerone, A., Turini, F.: Mining learning processes from FLOSS mailing archives. In: Janssen, M., Mäntymäki, M., Hidders, J., Klievink, B., Lamersdorf, W., Loenen, B., Zuiderwijk, A. (eds.) I3E 2015. LNCS, vol. 9373, pp. 287–298. Springer, Heidelberg (2015). doi:10.1007/978-3-319-25013-7_23
12. Paoletti, N., Yordanov, B., Hamadi, Y., Wintersteiger, C.M., Kugler, H.: Analyzing and synthesizing genomic logic functions. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 343–357. Springer, Heidelberg (2014). doi:10.1007/978-3-319-08867-9_23
13. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. **33**(1), 64–95 (2008)
14. Solar-Lezama, A., Rabbah, R.M., Bodik, R., Ebcioglu, K.: Programming by sketching for bit-streaming programs. In: Proceedings of PLDI 2005, ACM SIGPLAN Notices, vol. 40, pp. 281–294. ACM (2005)

15. Srivastava, S., Gulwani, S., Foster, J.S.: From program verification to program synthesis. In: Notices, A.S. (ed.) Proceedings of POPL 2010, vol. 45, pp. 313–326. ACM (2010)
16. van der Aalst, W.M.P., de Beer, H.T., can Dongen, B.F.: Process mining, verification of properties: an approach based on temporal logic, Beta Working Paper Series WT, p. 136. Eindhoven University of Technology, Eindhoven (2005)
17. van der Aalst, W.M.P., Stahl, C., Processes, M.B.: A Petri Net-Oriented Approach. The MIT Press, Cambridge (2011)