FORMULATING ASPECTS OF PAYPAL IN THE LOGIC FRAMEWORK OF *GBMF*

Min Li and Chris J. Hogger
Department of Computing, Imperial College London, U.K.

Keywords: GBMF, Business model, PayPal, Prolog.

Abstract: Logic-based modelling methods can benefit business organizations in constructing models offering flexible knowledge representation supported by correct and effective inference. It remains a continuing research issue as to how best to apply logic-based formalization to informal/semi-formal business modelling. In this paper, we formulate aspects of the general business specification of PayPal in logic programming by applying this in logic-based GBMF which is a declarative, context-independent, implementable and highly expressive framework for modelling high-level aspects of business. In particular, we introduce the primary PayPal business concepts and relations; specify simple but essential PayPal business processes associated with a knowledge base, and set core business rules and controls to simulate the PayPal case in a fully automatic manner. This specific modelling method gives the advantages of general-purpose expressiveness and well-understood execution regimes, avoiding the need for a special-purpose engine supporting a specialized modelling language.

1 INTRODUCTION

Business changes fast, so do business rules/logic. No company can guarantee an ever-effective structure which requires no further change. The research methods on how best to model or describe a business organization should also advance with the times. However, the majority of business models in the market are comparatively weak in dealing with fast-changing businesses, because a certain number of them are purpose-built with a short lifetime. In this circumstance, we believe it is necessary to abstract the commonality of various businesses with different types and scales, and then build a generic modelling framework, on which any given business could be further specified and modelled. We also argue that an admissive formal method should be used to build this framework in order to make it extensible, fault-tolerant, easy to understand, and expressive.

Our related work in the field of formal business modelling focuses on conceptual and logical aspects of business of a general nature. As the foundation of our work, we established a Generic Business Modelling Framework (named GBMF) which is mostly declarative, implementable, context-independent, and of high expressiveness for modelling high-level aspects of business. In the implemented GBMF, business plans and action sets, dynamically spawned processes, ontological variables and manageable assets with rich information are expressible transparently using an economical repertoire of primitive constructs, without requiring overly-burdensome programming effort. To better demonstrate the beauty of GBMF and its expressive power, we introduce the PayPal case study. We firstly define those core business plans aiming for personal users and corresponding action knowledge base to interpret those plans. We also build a small number of controls which are required to ensure the main program continuously and automatically output a certain amount of data for analyzing business further.

The contributions of this paper can be summarized as follows:

- It introduces GBMF, an all-round framework for representing diversified generic business entities and their relations;
- It implements an executable simulator upon GBMF to simulate given business instances in a automatic way;
- It applies the PayPal case to GBMF. The further analysis enables users to retrieve essential information from ontological variables and assets.

Li M. and Hogger C. (2009).
FORMULATING ASPECTS OF PAYPAL IN THE LOGIC FRAMEWORK OF GBMF.
In *Proceedings of the 11th International Conference on Enterprise Information Systems - Artificial Intelligence and Decision Support Systems*, pages 39-44
DOI: 10.5220/0001858600390044

In section 2 we discuss the related work. Section 3 outlines the basic structure and key concepts of GBMF. We introduce the PayPal business protocol in Section 4. Section 5 simply analyzes the implementation of PayPal simulation. Finally, in section 6 we present conclusions.

2 RELATED WORK

Existing business models vary from the abstract level, investigating macroscopic business concepts or general axiomatizations, down to the concrete level, working on context specific implementations.

At the abstract level, many recommendations exist as to the macroscopic concepts that need to be considered in modelling business (Fox, 1998) (Affua, 2003). There are clearly intersections and exclusions among these works, but ascertaining it precisely is difficult since not all of them are given sufficiently formal anchorage. Whilst there are many others intended for facilitating their design and implementation, categorized as the concrete level. Early exemplars include the business process modelling methods IDEF0 (NIST, 1993) and PSL (Schlenoff, 1997).

More recently, there emerges a community exposing detailed commitments to representation, formalization, logic and behaviours of business models, whose research can be classified as a mid-level in between abstract and concrete levels. From their views, concepts and behaviour formulation, logical transparency and expressivity power are all subtle factors for a business company in choosing their models. (Chen-Burger, 2002) expresses conditions and actions of business processes, relationships between them and constraints on the data they deal with. Gordijn proposes an ontology-based conceptual model named *e3value*, focusing on modelling conceptualization and e-business ontology, which is compared in detail by (Gordijn, 2005) with BMO (Osterwalder, 2005). The FBPML (Chen-Burger, 2002; Kuo, 2003), as a sophisticated amalgamation and extension of features drawn from PSL and IDEF3, is declarative, using logic to describe features of, and relations over, business processes. Although sharing similarities with GBMF's basic representations of actions, entities and process logic and behaviour, FBPML is a purpose-built language requiring its own custom-built engines and tools. GBMF is written directly in the general-purpose Prolog language and so freely inherit all the representational and execution power of that formalism, including the well-understood model semantics of normal-clause logic.

If we view business modelling, particularly the formalization of modelling, in the context of AI, it is inevitable to refer to business analysis and design. Early approaches included the Vienna Definition Method (Bjorner, 1978) and Structured Analysis (Yourdon, 1989). GBMF borrows the idea in early system engineering to further explore the semantic foundation by performing reification and decomposition to layers with acceptable details. When facing the task of balancing deterministic controls over ontological variables and declarative controls expressed as business rules, a good reference is the classical "logic+control" interplay first emphasized by (Kowalski, 1979).

In short, inspired by most of the state-of-art research on generic business modelling, GBMF not only represent and simulate high-level business to help users to better understand it, but also serve as a formal and simple specification of a prototype-supported method for quick business modelling.

3 GBMF

In principle, the business of an enterprise can be formulated as a purely declarative theory expressing various business entities, their properties, inter-relationships and controls (Hogger, 2004). Achievable goals of the business can then be identified with logical consequences of the theory, and derivations of those goals can be interpreted as particular simulations of the enterprise. A more practical approach is to replace parts of that theory by business plans and associated interpretation which, though still expressed declaratively, are inherently more deterministic to the extent that they embody some preconceived commitments to the control and interdependence of events.

3.1 An Overview of GBMF

Based on the above ideas in design, GBMF is built upon the general notion of activities operating upon any typical business entities. Activities are composed from atomic basic actions organized into action sets, which are in turn organized into larger programmatic hierarchies called business plans. A plan might, for example, embody the actions entailed in a production process from inception to delivery, with attendant impacts on strategy, financial and temporal aspects of the business. During manipulation of a business, such a plan could be applied on multiple occasions, possible concurrently. GBMF therefore treats a plan as a

template capable of spawning distinct instances called processes, each acting upon its own vector of business entities.

Many entities in a process will have a transient existence, being only intermediates for creating the eventual deliverables of that process. Those entities to which this does not apply are the deliverables that must survive, referred to as business assets. Thus the macroscopic behaviour of an executing GBMF instance is the transformation of an asset space, when various processes are dynamically spawned, possibly exploiting existing assets, copying or acquiring its required assets and creating new assets.

The instigations and progressions of processes are governed by business process rules, whilst the internal relationships between their entities are governed by the underlying procedures that define the basic actions. A well-structured GBMF instance should take a general form of self-controlling its progression by consulting the asset space about the required assets in advance. But in practice, additional controls are usually required in order to ensure that business process automation can be simulated in a more reasonable and interactive way.

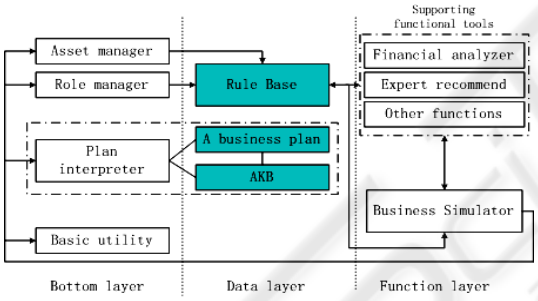

Figure 1: Overview of GBMF.

Figure 1 shows a structural view of GBMF. There are three layers involved in modelling a business with GBMF. The bottom layer contains modules to deal with generic assets, to manage role relations, and to parse business plans. independent of concrete business data, serving as an interpretation to make low level Prolog understand GBMF and execute given business plans smoothly. The bottom layer is treated as a minimum GBMF in semantic, based on which we develop a function layer to enhance the power of GBMF, in terms of managing functional assets and implementing advanced functional supporting tools.

In between these two layers, there is a data layer in which a modeller needs to provide concrete business data as GBMF input. GBMF can also be viewed as interdependency of functional sub-modules from a pure business perspective as described in (Li, 2007).

3.2 Plans and Processes

GBMF represents each basic action as a term of the form *Action-name(Ontvars)* in which *Ontvars* is a vector of ontological variables. Each basic action *A* appears within an action declaration whose general syntax is as:

```
    action(R, S, I, X).
or  action(R, S, I, C, X)
or  action(R, S, I, C, X1, X2)
```

R represents the role holder of this action, *S* names an action set and *I* is a position index for *A* within *S*. Each of *X*, *X1*, *X2* is a basic action or an action set name and *C* is a predicate, over one or more ontological variables, expressing a condition.

```
% plan "register"
action([accountadmin],register,1,
 copy(accountlist, t1)).
action([accountadmin],register,2,
 checkacv(newemail,accountlist,tag, t2)).
action([accountadmin],register,3,yes(tag),
 doregister(newemail)).
.....
```

Figure 2: The 'register' plan.

Figure 2 shows a fragment of the 'register' business plan formulated from the PayPal data file. The ability of one action set to invoke others inherently organizes a complete business model into a set of plans. A plan comprises a root action set, being invokable by no other, together with all those other action sets that it may invoke directly or indirectly. Some act sets could be repeatedly invoked by other plans, to economize on the use of common knowledge. Each action set has an associated control declaration which is either of the assertions *control(A, seq)* or *control(A, con)*. This specifies whether the basic action *A* is to be performed sequentially or concurrently. The former case uses the action declarations' indices to determine the temporal order, whilst the latter case ignores them.

A GBMF process is an executing instance of a plan. At any time in the animation of a model there may exist zero or more active instances of each plan, at various stages in their executions. A process is denoted by P_i where *P* is the plan name and *i* is a unique instance identifier. P_i has its own binding environment $\beta(P_i)$ containing a pair (*V*, *Val*) for each $V \in ont(P)$ signifying that *V* is *bound* to the value *Val*. As P_i executes, its variables become

instantiated by various ways - clock-binding, action performing and constraint evaluation. Performing *A* entails consulting an Action Knowledge Base (AKB) containing an associated procedure for each basic action type. If *A* is user-defined then its procedure will have been supplied in the AKB by the modeller. The primary effect of executing the procedure is to update $\beta(P_i)$.

3.3 Assets

By default, the termination of a process would leave no trace of its prior existence, since its bindings will be automatically garbage-collected. Instances of relations between its ontological variables would have been constructed or verified by the effects of actions and constraints, but would not survive to the lasting benefit of the modelled business as a whole.

In order to enable processes to manipulate business entities of greater permanency, GBMF allows the modeller to declare for any basic action type that some of its arguments denote durative assets. Concretely, such an asset is a value *Val* - a compound structure conforming to a schema declared in the AKB for any particular asset type, tagged with a unique identifier, a type, a status *S* and an origin. The general schema of an asset is:

```
asset(Id,Name,Class,Type,Val,S,Origin)
```

This schema makes it possible for the modeller to represent any meaningful entity as an asset by freely defining its value *Val* in AKB. The status of the asset is either public or process-owned. Its *origin* identifies the process π_0 from which it originated. Besides the action-defining procedures, the AKB contains asset declarations specifying any asset-handling entailed in each action type, which can add value to the fine controls over assets management. For example, it will help to infer plan-asset dependency or actionset-asset dependency.

Assets may serve many purposes, including message-passing, information-displaying, process-triggering etc. Our last remark about assets is that if any assets are required to survive through its serving process π's termination then π must beforehand make them public, by performing a system-defined basic action 'publish'. When a complete process terminates, what survives is the set of public assets still remaining in the asset space. These are the only observable deliverables of the real time processes.

3.4 Plan-asset Dependency

The plans and the AKB's asset declarations in a model induce a plan-asset dependency which can be treated as the fundamental logic to drive a business progression in GBMF. Plan-asset dependencies are logical consequences of the model and can be inferred by the asset manager based on asset declarations, however it serves the modellers' interests to assert them explicitly in a component of the model called the Business Process Rulebase (BPR), independent of the business data. They contribute to the formulation of business process rules regulating process creation and behaviour.

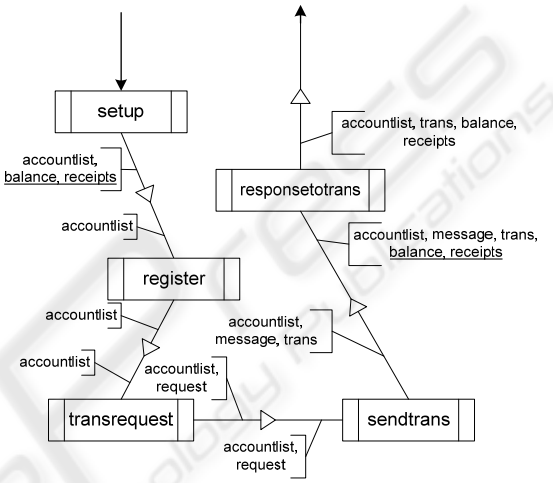

Figure 3: Plan-asset dependency graph.

The plan-asset dependency can be demonstrated as a graph whose nodes are plan names. Figure 3 outlines a small fraction of the plan-asset dependency graph for the PayPal business model, which is indeed a single thread of a successful sending transaction. In Figure 3 two solid arrows represent the entry and exit of a single money-send thread. Each edge directed from one plan *P* to another *Q* is labelled by two sets of variables in *ont*(*P*), in which the set at the upper end of hollow arrows are *P*'s produced assets, the other at the lower end indicate those assets required by *Q*. For example, the 'setup' plan produces three assets in which only *accountlist* is required by its subsequent plan 'register'. The other two have a free existence in the asset space for further use by other plans.

More generally and importantly, the BPR's process rules can express any user-defined controls about the behaviours of the process pool if these are expressible as logical conditions over existing processes, or their binding environments, or the existing assets. They are consulted by the model's execution manager to drive the model forwards and to ensure that each process is spawned to serve a declared purpose and that its subsequent behaviour

satisfies any declared conditions. The user-defined controls and the intrinsic plan-asset dependencies are both stored in a rule base.

4 PROTOCOL OF PAYPAL

The PayPal service allows a customer to pay in various ways, including through credit cards, bank accounts, buyer credit or account balances, without sharing financial information (PayPal.com). The popularity, the simplicity and the conveniences it can bring to online payment are reasons why we choose PayPal as one of our case studies.

In practice, we are not able to exploit a complete business of PayPal because some derivative business processes, designed for company customers, are hard to obtain. However, there are some primary business processes of PayPal for normal customers, serving as our main target in this paper.

Based on (Burchell, 2004) and our practices, we describe the business protocols of PayPal as follows:

1. Two roles are involved: user and server. We further distinguish user into sender and receiver, server into general admin, account admin, transaction server and accountant.
2. All users should be authenticated by registering and logging into PayPal system before they use.
3. A sender can send a transaction request by providing minimum information such as sender id, receiver id, amount and payment method. Payment can be made through an existing account or a valid credit card. The request will be validated before it is set as 'pending' and stored into the transaction database. The requested money will then be debited from the sender's account and a message will be generated to notify the receiver.
4. After log in, a receiver can either accept, by crediting money into account and finishing the transaction, or reject a transaction by setting as 'rejected' and returning money to sender.
5. Senders can cancel their pending transaction by raising a cancel request. Any transaction older than 30 time units will be removed from the transaction database if it has not been claimed by receiver or not been cancelled by sender.
6. Senders can claim money within 30 time units from the completion date of the transaction if: the sender didn't claim on this transaction before; the sender has less than two claims in that year; the sender's claim has been approved by PayPal as legal by satisfying given claim conditions. If the claim is successful, the money will be transferred from receiver to sender.

PayPal business makes profit by charging a fee comprised of a proportion of transaction amount and a constant administration fee for every successful transaction. The primary functions involved in the above protocols make up the basic business knowledge for our PayPal case design. We also define some additional functions, such as transaction query for modellers in exploring more information through the PayPal simulation.

5 SIMULATING PAYPAL

A general view on how the PayPal business processes run under GBMF is shown in Figure 4. To clearly describe and model the primary PayPal business activities discussed earlier, we define 10 business plans in the data layer, represented by rectangles in the graph. The plan 'register' shown in Figure 2 is one of them in the form of source codes, conforming to the plan schema. The defined assets are listed in the cylinder in Figure 4, including three types of customer requests, detailed transaction information, several types of messages, PayPal's financial account balance, inventory of receipts and records for all registered user accounts.

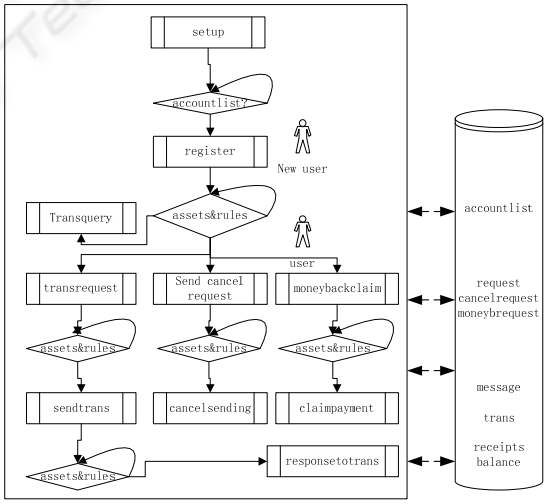

Figure 4: PayPal simulation in GBMF.

The simulation starts from 'setup', which is a special type of plan to initialize the PayPal business environment by clearing up the buffer and setting three empty assets, including the *accountlist*. Then the 'register' plan will actively look for *accountlist*, whether empty or not, by searching the asset space

in a given time. Once *accountlist* is found, it will be reserved and registered by the 'register' plan. Then 'register' will be triggered and executed, by inserting a new list into the asset *accountlist* and recording information of a new user. Along with the advance of the system clock, the main simulation will carry on. Thus, for all PayPal plans except 'setup', they will continuously and repeatedly check the asset space, competing for assets they need. At the same time, the additional trigger conditions defined by users will be evaluated, as further constraints. We use the tag *assets&rules* in diamonds to denote controls, including plan-asset dependencies and additional trigger conditions. The one-way arrows in Figure 4 generally indicate the ordinal relation between plans, conforming to plan-asset dependencies we discussed earlier.

PayPal simulator runs over a timer controlled by the execution manager. From bindings of temporal variables, users are free to view business at a specific time point or during a defined time interval. Insofar as we want to automate the whole simulation process as much as possible, we assume some information is given, such as registration details of a new PayPal user. Certain other information should be generated in run-time, such as various requests sent by customers in real-time.

Through the simulation of the PayPal use case, users can understand the basic logic of PayPal as to how different business processes compete for assets needed, how a new business process is triggered when all its required assets exist in asset space, how an active process produces assets required by others before they die with garbage-collecting, how the business make profit by charging an administration fee for some types of transaction. It is also possible for the modeller to achieve high-level business goals by either defining them in original business plans or implementing them in advanced functional modules in function layer.

6 CONCLUSIONS

GBMF facilitates business specifications by establishing a generic business modelling framework which offers logical formulations and reasoning mechanisms aiming to provide a high-level, transparent and flexible means of expressing the diverse entities and constraints typically encountered in business. Our case study in formulating aspects of the PayPal business demonstrates the rich expressiveness of GBMF in representing business activities and goals. It is also shown in PayPal business simulation that intrinsic plan-asset dependencies and user-defined controls can easily guild execution manager to manipulate various ontological entities including ontological variables and generated assets during the run time.

The ultimate purpose of GBMF is to provide an alternative modelling method with a sound logical structure and simple semantics, hence a synthesis to support business consultation, validation and prototype. The rich extensibility of GBMF enables modellers to develop advanced functions to support high level analysis for more complex business.

REFERENCES

Afuah, A., Tucci, C.L., 2003. *Internet business models and strategies.* McGraw Hill. Boston.

Bjorner, D. et al, 1978. *The Vienna Development Method: The Meta-Language*. LNCS Vol. 61, Springer.

Burchell, D., Nielsen, D., Sofield, S., 2004. *PayPal Hacks*, Publisher: O'Reilly.

Chen-Burger, Y-H., et al, 2002. Enterprise Modelling: A declarative approach for FBPML. In *ECAI: Workshop on Knowledge Management and Organizational Memories.*

Fox, M.S., Gruninger, M., 1998. Enterprise modeling. In *AI Magazine 19(3)*.

Gordijn, J., Osterwalder, A., Pigneur, Y., 2005. Comparing two business model ontologies for designing e-business models and value constellations. In *18th Bled e-Conference on e-Integration in Action.*

Hogger, C.J. and Kriwaczek, F.R. 2004. Constraint-guided enterprise portals, *Proc. of 6th Int. Conf. on Enterprise Information Systems*, pp. 411-418.

Kowalski, R., 1979. *Logic for Problem Solving*, North Holland Elsevier.

Kuo, H-L., Chen-Burger, Y-H., Robertson, D., 2003. Knowledge management using business process modeling and workflow techniques. In *IJCAI'03, Workshop on Knowledge Management and Organizational Memories.*

Li, Min and Hogger, C. J., 2007. Generic constraints-based framework for business modelling. In *Trends in Enterprise Application Architecture*, LNCS, Vol. 4473, Springer Verlag, pp.241-254.

NIST - National Institute of Standards and Technology, 1993. *Integration Definition for Function Modelling (IDEF0)*. NIST. Gaithersburg.

Osterwalder, A., Pigneur, Y., Tucci, C.L., 2005. Clarifying business models: origins, present, and future of the concept. In *Communications of the Association for Information Systems, 16.*

Schlenoff, C., Knutilla, A., Ray, S., 1997. In *Proceedings of the Process Specification Language (PSL) Roundtable.* NIST. Gaithersburg.

Yourdon, E., 1989. Modern Structured Analysis. Yourdon Press, Englewood Cliffs, New Jersey.