Multi-Agent Transactions for Web-based Design Activities

Muhammad Younas, Kuo-Ming Chao, Rachid Anane, Anne James
Data and Knowledge Engineering Research Group (DKERG)
School of Mathematical and Information Sciences, Coventry University, UK
{m.younas, k.chao, r.anane, a.james}@coventry.ac.uk

Abstract

Web-based engineering design activities require fault tolerance and concurrent access to shared resources such as databases and Web servers. Such activities are generally dynamic, cooperative, long-lived, interactive, and non-prescriptive. We propose a new multi-agent transaction model, which is based on extended transactions and multi-agent technologies. The novelty of this model is that it automatically customises transactions to the requirements of the design activities. In addition, this model is believed to improve concurrency and fault tolerance, facilitate interaction and co-operation of participating systems involved in design activities.

1. Introduction

Complex engineering design (e.g., offshore oil platforms, aeroplanes, ships) requires collaborative work among diverse design disciplines. Such design environments often involve a number of different organisations that specialise in different design disciplines, which are distributed across the network. The Web, an emerging design environment, provides an infrastructure for design activities to be geographically distributed among different teams and design systems. In such highly dynamic environment the nature of the work and the requirements may change continuously. More importantly, these activities require consistent and concurrent access to shared resources (e.g., databases, Web servers, etc.) and also require fault tolerance. This motivates the need for providing suitable techniques to facilitate this kind of cooperative activities.

Existing design approaches apply Extended Transaction Models (ETM) to design activities [8, 15, 6]. ETM relax isolation and atomicity properties of the ACID (atomic, isolated, consistent, durable) transactions [5]. ETM are used to provide design activities with concurrency and consistency aspects. However, one of the major problems with current ETM systems is the decision-making. For instance, it is difficult to determine where to split and join transactions, or where to compensate or replace transactions.

Similarly, multi-agents are also used in the design activities [13, 12]. The role of agents in design activities is to support communication and to improve the co-ordination among design systems. However, transaction-like facilities (e.g., ensuring concurrent and consistent access to shared resources [9]) are not well addressed in multiple design agent systems.

Our research identifies that the exclusive use of one of the above technologies (i.e., transactions and multi-agent) remains ineffective in the design activities. In this paper we aim to develop a new model that addresses the issues of Web-based design activities by combining multi-agent technology with ETM. One distinguishing feature of the proposed model is that it provides a facility for automatically customising a variety of ETM to the requirements of design activities. Existing approaches [6, 15, 1] require users to specify transaction models to suit their needs. In addition, the proposed model is believed to improve concurrency and fault tolerance, facilitate interaction and co-operation of participating systems.

The paper is structured as follows. Section 2 identifies requirements of the design activities. Section 3 reviews current approaches and establishes their limitations to design activities. Section 4 formally specifies our proposed model. Finally, section 5 concludes the paper.

2. Requirements

The requirements are based on the characteristics of designing large and complex engineering products (e.g., offshore oil platforms, aeroplanes, ships). Such environments often involve a number of different organisations that specialise in different design

disciplines. This requires multiple design systems to co-operate. The participating systems could be semi automated or fully automated design systems with data store facilities. They should be able to operate concurrently in order to optimise the use of required design resources. Such design activities are characterised by long duration, co-operation and negotiation, non-prescriptive developments, and recovery from failures. We discuss these characteristics within the context of the requirements of Web design activities so as to define the scope for this research.

i). Long Duration

Transactions in design environment are generally of long duration [6]. These transactions may run for hours or days to process the design activities. In particular the activity may require deliberation from humans. It is difficult to determine in advance the execution time of such transactions. Thus it is required to ensure that a particular data object is not blocked unnecessarily for the duration until a transaction is committed. For example, in designing an offshore oil platform, the cost engineer issues a design change that requires a semi-automated process flow design system and an automated layout design system to respond the change [3]. The layout design system may take a few minutes to propose a new layout, but the process flow designer may take days to reflect the change. Thus, the layout system cannot commit the transaction until the process flow designer completed the change. Thus it is required to commit the completed part of the transaction so as to ensure the early release of system resources.

ii). Co-operation

Co-operation among the participating systems of the design activities is a prerequisite for the accomplishment of various tasks involved in design activities [7, 6]. For such systems it is necessary to share different data objects of the design activities. Thus mechanisms are required to facilitate the exchange of intermediate results, while at the same time guaranteeing that consistency of these results is maintained during exchanges. For example, the safety and layout systems might be working on a location and orientation of a hazardous equipment item such as a compressor. This requires both parties to modify two parts of the same data object concurrently, with the intent of integrating these parts to create a new version of the design. In this situation, the systems might need to look at each other's work to ensure that they are not modifying two parts in a way that would lead them to have conflicts.

Furthermore, in design activities participating systems need to cooperate so as to achieve a common goal. Moreover, for some design activities, it is necessary that participating systems develop a temporary cooperation that may exist for a particular task, or for a limited time. Such temporary cooperation needs to be formed dynamically with different participating parties and in order to meet different requirements.

iii). Non-prescription

Design activities are generally interactive and non-prescriptive. Non-prescription implies that the system can not determine in advance the nature of the transactions involved in design activities. That is, the system may not determine in advance that the execution of transactions need cooperation, or they might violate consistency of data objects, except by actually executing them. For example a generic connector can be used to connect various families of valves, pipes, and other equipments. The connector can be designed in the first place without considering specific applications. However, it can be used whenever it is required. Thus to support non-prescriptive design tasks, the system should be able to start a transaction, interactively execute operations within it, dynamically restructure it, if needed, and commit or abort it at any time.

iv). Failure Recovery

Various causes contribute to Web failures such as Internet communication failure, systems failures (e.g., Web servers), etc. These failures severely affect the design activities by interrupting their progress. Such interruption may result in frequent roll back. However, frequent roll back of the long-lived design transactions is not acceptable as a valuable piece of completed work might be lost. Instead, a transaction should be able to proceed (and eventually succeed) even if it partially completes. For example, the process flow engineer may propose a high cost equipment (e.g., condensate vessel) which may not meet the cost engineer's requirement. Thus the roll back should only occur for the part of the transaction in the process flow design, but not in the layout solution as it may still be valid.

3. Related Work

Classical database transactions strictly follow the ACID (atomic, isolated, consistent, durable) correctness criteria [5]. However, these criteria remain inappropriate for design activities. In particular their isolation and atomic policy does not suit the characteristics of design activities — which are cooperative, dynamic, long-lived,

interactive, and non-prescriptive. To overcome the limitations of ACID transactions, numerous extended transaction models (ETMs) have been proposed [5, 8, 15, 6, 10] for the design activities. ETM generally focus on the relaxation of isolation and atomicity properties. In [15] an ETM-based system, called Jpernlite, is developed for Web-enabled software engineering applications. Jpernlite mainly concerns the concurrency control mechanism of designed activities. However, it does not fully support the characteristics of design activities. For example, no dynamic restructuring of transactions is supported. Support for dynamic restructuring of transactions in design activities is provided through Split and Join transaction models [6, 10]. The former splits an active transaction, T_i, into multiple transactions, T_i and T_j, by delegating the actions of T_i to T_j (T_i is assumed to preserve its identity after the split). T_i and T_j can commit or abort independently. The latter merges multiple independent transactions, T_i and T_j, into a single transactions, T_k.

In addition to transaction technology, multi-agent systems (MAS) are also used in the design activities [13, 3]. An agent is an intelligent and autonomous system that can control its own behaviour and respond to the requests from other agents in order to achieve a common goal [12]. An agent incorporates a reasoning mechanism such as Belief Desire and Intention (BDI) model — which is a procedural reasoning method that enables agents to deal with highly dynamic activities such as design. An agent can be aware of its environment and can refer to its own goal to select appropriate plans and carry out related actions in order to reach a desired state. Thus, the BDI model provides agents with a mechanism for representing knowledge (beliefs) and for modelling its behaviour (semantics). A dynamic environment often incorporates unexpected situations. It is therefore necessary for agents to handle exceptions so as to ensure that the system is consistent and it avoids abnormal termination.

The role of agents in design activities is to support communication and to improve co-ordination among design systems. The consequence is to present an integrated environment for the exchange of information and knowledge between participating design systems. However, transaction-like facilities (e.g., ensuring concurrent and consistent access to shared resources [9]) are not well addressed in multiple design agent systems.

As stated earlier that exclusive use of the above technologies (i.e., transaction or multi-agent systems) remains ineffective in the design activities. One of the major problems with current ETM systems [15, 10] is the decision-making facility. For instance, it is difficult to determine where to split and join transactions, or where to compensate or replace transactions. Furthermore, some activities may trigger other activities that may require cooperation from different heterogeneous design disciplines (e.g., in offshore oil platform). These disciplines may need to cooperate and negotiate before performing the requested tasks that pertain to designed activities. For example, a pump needs to be upgraded in order to produce bigger pressure than the current one. This implies that the size of the pump will increase, so the layout design system needs to be partially re-designed. However, the proposed pump may require excessive usage of electricity, which the current power generator cannot supply. Thus, the change of the power generator is inevitable, if the new pump is to be installed. In such environment, activities are non-prescriptive and thus it is difficult to model them in advance. This requires understanding the context and well-defined semantics. Agents have the capability to facilitate decision-making, provide cooperation and negotiation between different heterogeneous disciplines, and model non-prescriptive activities.

The above discussion makes it imperative to combine multi-agent and transaction technologies so as to provide newer services in different application domains. Interestingly, combining multi-agent with transaction technologies has been investigated in [2, 4, 8, 11]. However, [2, 4, 8] mainly concern e-commerce applications. For example, [4] uses multi-agents to model dynamic Web transactions in the e-commerce applications, and introduces a scripting language, called Multi-Agent Processing Language (MAPL), for the execution and interaction of agents. Similarly, in [2] agent technology is applied to model the cooperative transactions in e-commerce applications. Furthermore, [8] applies transaction techniques to multi-agent systems in scheduling production orders in a manufacturing environment. It applies open-nested transactions to schedule different requests made for a particular product. This approach also corresponds to the e-commerce applications. Moreover, [11] has laid down some requirements for multi-agent transactions in design activities. However, this approach is limited to the definition of requirements as no solution is proposed.

4. Multi-Agent Transaction Model for Web-based Design Activities

This section describes a new model that addresses the issues of Web-based design activities. Compared with others our approach is unique in that we combine multi-agent technology with advanced transaction processing techniques within Web-based design activities.

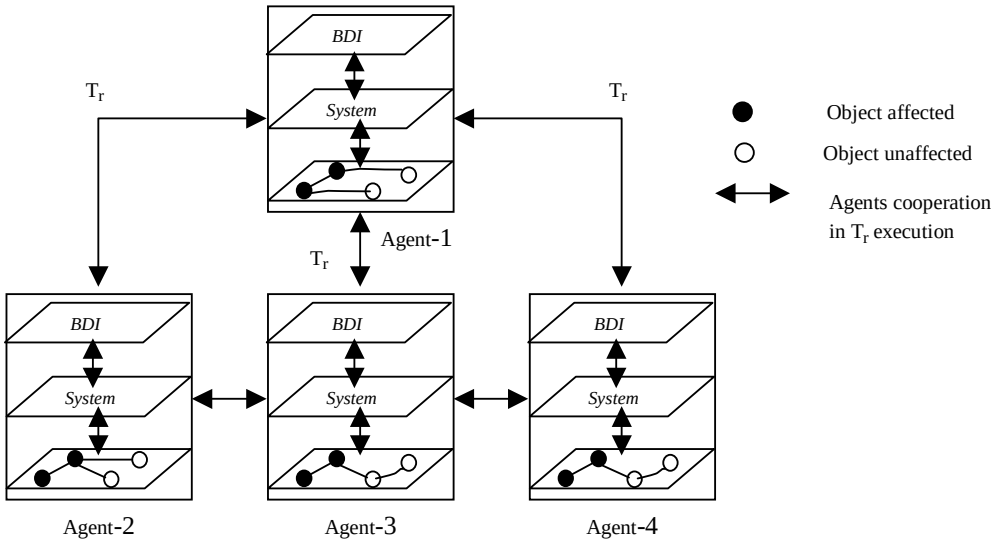

Figure 1. A Generalised Architecture for Multi-Agent Transactions

The proposed model is a combination of multi-agents [13, 12] with well-defined Extended Transaction Models (ETM) such as split/join, open-nested transactions [14, 6, 10]. An interesting feature of the proposed model is that it provides a facility to automatically specify a variety of ETM that match the needs of a particular design activity. On the other hand, existing approaches [15, 1] require the users to specify the transaction models to suit their needs. In addition, the new model is believed to: improve concurrency and performance by avoiding resource blocking, increase availability and failure recovery, facilitate the interaction and co-operation among the participating systems of design activities.

The proposed model is diagrammatically shown in Fig. 1, in which multiple agents are used to execute the transactions of the design activities. An agent is called a *base agent* (denoted $agent_B$) if it starts the execution of the root (or main) transaction (denoted T_r). Other agents are called participating agents. A set of participating agents (e.g., Agent-2,3,4) is represented as $agent_P$. A particular participating agent is represented as $agent_{Pi}$ (i = 2,3,4). In Fig. 1, Agent-1 is a base agent ($agent_B$) as it starts executing root transaction T_r. T_r is executed to perform a particular design activity (denoted D_{act}).

In this model agents (i.e., $agent_B$, $agent_P$) are used to automatically customise a variety of ETM for the execution of T_r so as to match the needs of a particular D_{act}. Agents customise ETM using their BDI model, which enables an agent to select appropriate plans and in order to reach a desired state [12]. Thus depending on the nature of D_{act}, agents can automatically specify a particular ETM for the transactions involved in D_{act}. Automatic customisation of ETM (for $T_r \in D_{act}$) is performed (by $agent_B$, for example) through the following actions A1-A5 (note that $agent_P$ can also take these actions):

A1. $agent_B$ requests $agent_P$ to cooperate in the execution of T_r
A2. during the execution of T_r, $agent_B$ splits T_r into independent transactions (T_i and T_j) which can be executed by $agent_B$ and/or $agent_P$
A3. before the execution of T_r, $agent_B$ divides T_r into different subtransactions ($T_{r1}…T_{rn}$) which can be executed by $agent_B$ and/or $agent_P$
A4. before/during the execution of T_r, $agent_B$ associates alternative and compensating transactions with the subtransactions of T_r
A5. during the execution of T_r, $agent_B$ joins T_r with another independent transaction, T_j, thus forming a new transaction T_k.

The basic technique used by agents in exercising actions (A1-A5) is to determine dependencies between the transactions of D_{act}. Various dependencies occur between different independent transactions or within the same transaction of D_{act}. Following are some example dependencies [16]:

- Two transactions T_i and T_j are said to be dependent if they access a shared data object, obj. Such dependency is formally modelled as follow.

  $$T_i\,(obj) \wedge T_j\,(obj) \Rightarrow (T_i\ \boldsymbol{DD}\ T_j)$$

  where, ***DD*** shows data dependency between T_i and T_j

- T_i and T_j are dependent if T_i and T_j can be merged together to produce a single transaction, T_k.

  $$\forall\ 0 \leq i, j \geq n\ (T_i\ \boldsymbol{JD}\ T_j) \Rightarrow T_k$$

  where, ***JD*** shows join dependency between T_i and T_j

- T_i and T_j are dependent if the commit of T_i depends on the commit of T_j.

  $$\forall\ 0 \leq i,j \geq n\ (T_i\ \boldsymbol{CD}\ T_j) \Rightarrow commit(T_i) \wedge commit(T_j)$$

  where, ***CD*** is commit dependency between T_i and T_j

Based on the dependencies, agents execute transactions according to a required ETM. Agents use dependencies to communicate with each other in order to execute transactions of D_{act}. For example, $agent_B$ may need $agent_P$ (as in A1) to cooperate in T_r. That is, part of T_r can be executed by $agent_P$ (but before the execution of T_r $agent_P$ needs to evaluate the impact of T_r on them. This is done via BDI and underlying mechanisms).

If T_r needs splitting (as in A2) then agents execute T_r according to the split transaction model [6]. That is, if $agent_B$ determines that some parts of the transaction, T_r, become independent from the rest of T_r then split operation is applied to $T_{r,}$ which splits T_r into multiple independent transactions, e.g., $T_{r,}$, T_i, and T_j. Thus new transactions (e.g., T_i, T_j) may be executed by $agent_P$. The advantage of the split is to increase concurrency by ensuring the early commit of the completed parts of T_r in D_{act}. Similarly, if T_r needs to be joined with other transactions (as in A5) then a join transaction model will be followed [6]. That is, if $agent_B$ believes that T_r and some other transaction T_j ($\in D_{act}$) have join dependency, then they can be merged into another transaction, T_k. Advantage of merging T_r and T_j into T_k, is that T_k may perform D_{act} efficiently as compared to the independent execution of T_r and T_j. Further, if T_r needs to be divided into different subtransactions (T_{r1}…T_{rn}) or to be associated with alternative/ compensating transactions (as in A3, A4) then a closed/open nested transaction model [5, 14] will be followed. That is, $agent_B$ and $agent_P$ decide which part of the transaction (or subtransaction) needs to be compensated or replaced with other transactions. Advantages of defining alternative transactions are to increase fault tolerance and service availability. Similarly, the use of open-nested transactions improves concurrency by ensuring the unilateral commit of subtransactions [5,14].

Moreover, agents maintain consistency by negotiating with each other. That is, before executing their transactions, agents have to determine dependencies between them so as to know whether a change made by a transaction of one agent ($agent_B$) has any affect on others ($agent_P$).

The above model provides a framework within which transactions can be automatically customised to the needs of Web-based design activities. Currently we are working on the implementation of the proposed model. In order to fully evaluate the potential contributions of our model, an industrial case study is required, which is a part of the future research.

5. Conclusion

In this paper we have highlighted that transactions of design activities are highly dynamic, cooperative, long-lived, interactive, and non-prescriptive. Moreover, these activities require fault tolerance and consistent and concurrent access to shared resources (e.g., databases, Web servers). These characteristics require that transactions must be dynamically managed according to the ETM (extended transaction model) that suit the respective design activity.

We have therefore proposed a novel multi-agent transaction model, which is based on ETM and multi-agent technologies. The new model is formally defined, which provides a framework within which transactions of the Web-based design activities can be modelled according to different ETM that suit the needs of design activities. This model is believed to improve concurrency and fault tolerance, facilitate interaction and co-operation of participating design systems within the Web environment.

6. References

[1] A. Biliris, S. Dar, N. Gehani, H.V. Jagadish, K. Ramamritham “Asset: A System for Supporting Extended Transactions” *ACM SIGMOD*, May 1994, 44-54.

[2] Q.Chen, U. Dayal "Multi-agent Cooperative Transactions for E-Commerce" *Conf. on Cooperative Information Systems*, 2000.

[3] Chao, K-M, Norman, P, Anane, R., James, A. " An agent based approach for Engineering Design" *Journal of Computers in Industry*, Vol 48, No 1, 2002, pp 17-28.

[4] Sylvanus A. Ehikioya. "An Agent-based System for Distributed Transactions: A Model for Internet-based Transactions" *IEEE Canadian Conf. on Electrical and Computer Engg*., Edmonton, Alberta, Canada, May, 1999

[5] A.K.Elmagarmid "Database Transaction Models for Advanced Applications" *Morgan Kaufman, 1992*.

[6] G.E. Kaiser "Cooperative Transactions for Multi-User Environments" *in Won Kim (ed.), Modern Database Systems: The Object Moodel, Interoperability and Beyond*, ACM Press, 1994, 409-433

[7] M. Mock, M. Gergeleit, E. Nett "Cooperative Concurrency Control on the Web" *Proc. of 5th IEEE Workshop on Future Trends of Distributed Computing System*, Tunis, October 1997.

[8] K. Nagi "Transactional Agents: A Robust Approach for Scheduling Orders in a Competitive Just-in-Time Manufacturing Environment" *Proc. of Workshop on MAS in Logistic and Economical Perspectives of Agents on Conceptulisation*, Germany, Sept., 1999

[9] E. Pitoura "Transaction-based Coordination of Software Agents" *9th Int. Conf., DEXA*, 1998

[10] C. Pu, G.E. Kaiser, N. Hutchinson "Split Transactions for Open-Ended Activities" *In proceeding of 14th VLDB Conference*, 1988.

[11] H. Ramampiaro, M. Divitini, S.A. Petersen "Agent-based Groupware: Challenges for Cooperative Transaction Models" *Proc. of the International Process Technology Workshop* (IPTW), Grenoble, Sept., 1999.

[12] Rao, S. A., & Georgeff. M. P. "BDI Agents: From Theory to Practice" *Proc. of 1st international Conference on Multiple Agent System*, 1995

[13] W. Shen **"**Distributed Manufacturing Scheduling Using Intelligent Agents" *IEEE Intelligent Systems*, Vol. 17, No. 1, 2002, pp 80-94,

[14] M. Younas, B. Eagelstone, R. Holton "A Review of Multidatabase Transactions on the Web: From the ACID to the SACRED" *Proc. of British National Conference on Databases (BNCOD)*, Exeter, UK, Springer, LNCS, 2000

[15] J.Yang, G.E.Kaiser, "JPernLite: An Extensible Transaction Server for the World Wide Web" *IEEE Transaction on Knowledge & Data Engineering*, 1999 (639-657)

[16] P.K. Chrysanthis, K. Ramamritham "Synthesis of Extended Transaction Models using ACTA" *ACM Transaction on Database Systems*, Vol. 19, No. 3, September 1994, Pages 450-491