A Multiagent Architecture Addresses the Complexity of Industry Process Re-engineering

John Debenham
University of Technology, Sydney
debenham@it.uts.edu.au

Abstract. Industry processes are the trans-corporate business processes required to support the e-business environment. Industry process re-engineering is the re-engineering of trans-corporate business processes as electronically managed processes. Industry process re-engineering is business process re-engineering on a massively distributed scale. Industry processes will not be restricted to routine workflows that follow a more-or-less fixed path; they will include complex processes for which their future path may be unknown at each stage in their existence. So a management system for industry processes should be both highly scalable and should be able to deal with such complex processes. A multiagent process management system is described that can manage processes of high complexity. This system is built from interacting autonomous components so achieving system scalability.

1. Introduction

Business to Business (B2B) e-commerce is driving a new generation of Internet applications that can dramatically automate trans-corporate industry processes only if the business systems and data that drive these industry processes are integrated across the component organisations [1]. Here *industry processes* includes both business transactions, and business processes and workflows. These Internet applications can only automate industry processes if there is a method to describe collaborative processes across organisations and to provide data interoperability. Improvements in process management can only be achieved through automation. Automation of processes leads to faster cycle times, reduced overhead and more competitive offerings. *Industry process re-engineering* is the re-engineering of trans-corporate processes as electronically managed processes. Companies that have implemented this e-business vision are saving tens of millions of dollars per year [2]. Industry process re-engineering must address the four issues of complexity, interoperability, communication and management:

- *complexity* of industry processes refers to their nature which includes *all* trans-corporate processes from routine workflows to high-level emergent processes [3];
- *interoperability* is an issue due to the heterogeneity of the diverse systems across a trading community. These systems vary in the applications that manage them and in the data formats that they employ;

- the *communication* and messaging infrastructure chosen will operate in a mission-critical environment, and
- process *management* which is responsible for tracking the automated trans-corporate processes that may include processes unique to individual trading partners and will probably involve a wide range of process steps that must "make sense" to all involved.

That is, industry process re-engineering must deliver a secure, scalable and reliable solution for running a company's most critical core business processes. The complex nature of industry processes is considered here.

High-level emergent processes are business processes that are not predefined and are ad hoc. These processes typically take place at the higher levels of organisations [1], and are distinct from production workflows [2]. Emergent processes are opportunistic in nature whereas production workflows are routine. How an emergent process will terminate may not be known until the process is well advanced. Further, the tasks involved in an emergent process are typically not predefined and *emerge* as the process develops. Those tasks may be carried out by collaborative groups as well as by individuals [4]. For example, in a manufacturing organisation an emergent process could be triggered by "lets consider introducing a new product line for the US market".

From a process management perspective, emergent processes contain "knowledge-driven" sub-processes and conventional "goal-driven" sub-processes [5]. The management of a *knowledge-driven process* is guided by its 'process knowledge' and 'performance knowledge' and *not* by its goal which may not be fixed and may mutate. On the other hand, the management of a *goal-driven process* is guided by its goal which is fixed, although the individual corporations involved in an industry process may not achieve such a fixed goal in the same way.

Multiagent technology is an attractive basis for industry process re-engineering [6]. A multiagent system consists of autonomous components that interact with messages. The scalability issue is "solved"—in theory—by establishing a common understanding for inter-agent communication and interaction. Specifying an inter-agent communication protocol may be tedious but is not technically complex. Standard XML-based ontologies will enable data to be communicated freely [1] but much work has yet to be done on standards for communicating expertise. Specifying the agent interaction protocol is a more complex as it in effect specifies the common understanding on the basis of which the whole system will operate. A multiagent system to manage "goal-driven" processes is described in [7]. In that system each human user is assisted by an agent which is based on a generic three-layer, BDI hybrid agent architecture. The term *individual* refers to a user/agent pair. That system has been extended to support knowledge-driven processes and so to support emergent process management and the full range of industry processes. The general business of managing knowledge-driven sub-processes is illustrated in Fig. 1, and will be discussed in Sec. 2. Any process management system should address the "process knowledge" and the "performance knowledge. *Process knowledge* is the wisdom that has been accumulated, particularly that which is relevant to the process instance at hand. *Performance knowledge* is knowledge of how effective people, methods and plans are at achieving various things. Sec. 3 discusses the management of the process knowledge. Sec. 4 describes the performance knowledge which is communicated between agents in contract net bids for work. Sec. 5 describes the agent interaction mechanism.

2. Industry processes

Following [8] a *business process* is "a set of one or more linked procedures or activities which collectively realise a business objective or policy goal, normally within the context of an organisational structure defining functional roles and relationships". Implicit in this definition is the idea that a process may be repeatedly decomposed into linked sub-processes until those sub-processes are "activities" which are atomic pieces of work. [viz (op.cit) "An *activity* is a description of a piece of work that forms one logical step within a process."]. A particular process is called a (process) *instance*. An instance may require that certain things should be done; such things are called *tasks*. A *trigger* is an event that leads to the creation of an instance. The *goal* of an instance is a state that the instance is trying to achieve. The *termination condition* of an instance is a condition which if satisfied during the life of an instance causes that instance to be destroyed whether its goal has been achieved or not. The *patron* of an instance is the individual who is responsible for managing the life of that instance [9]. At any time in a process instance's life, the *history* of that instance is the sequence of prior sub-goals and the prior sequence of knowledge inputs to the instance. The history is "knowledge of all that has happened already".

From a process management viewpoint, industry processes can be seen as consisting of sub-processes that are of one of the three following types:

• A *task-driven process* has a unique decomposition into a—possibly conditional—sequence of activities. Each of these activities has a goal and is associated with a task that "always" achieves this goal. Production workflows are typically task-driven processes.

• A *goal-driven process* has a process goal, and achievement of that goal is the termination condition for the process. The process goal may have various decompositions into sequences of sub-goals where these sub-goals are associated with (atomic) activities and so with tasks. Some of these sequences of tasks may work better than others, and there may be no way of knowing which is which [3]. A task for an activity may fail outright, or may be otherwise ineffective at achieving its goal. In other words, process failure is a feature of goal-driven processes. If a task fails then another way to achieve the process goal may be sought.

• A *knowledge-driven process* has a process goal, but the goal may be vague and may mutate [10]. Mutations are determined by the process patron, often in the light of knowledge generated during the process. After performing a task in a knowledge-

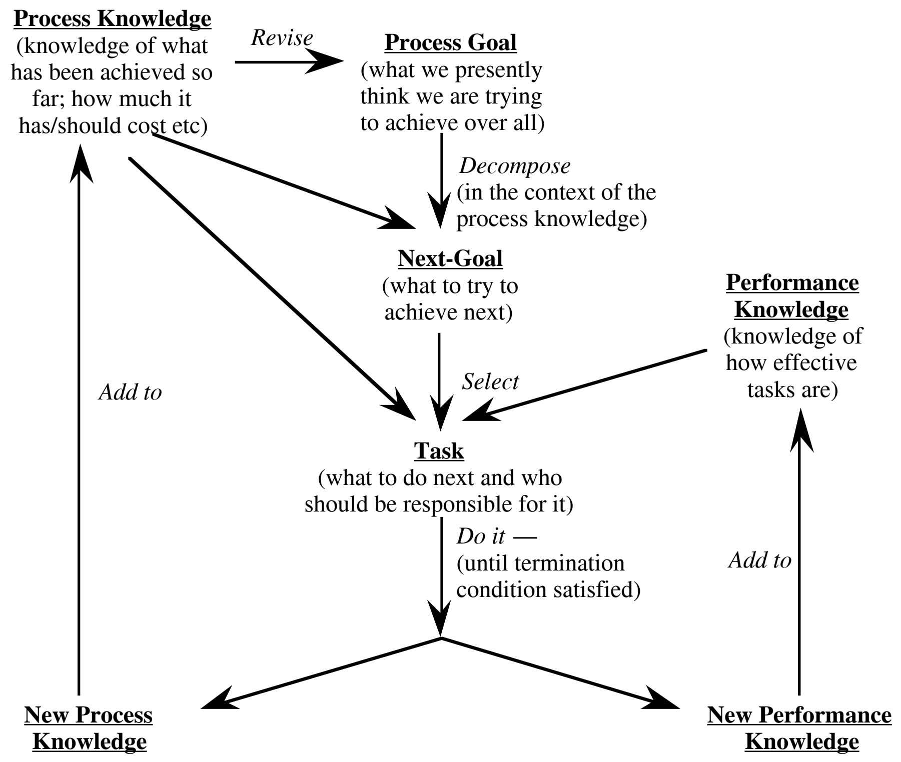
Process Knowledge
(knowledge of what has been achieved so far; how much it has/should cost etc)
Revise
Process Goal
(what we presently think we are trying to achieve over all)
Decompose
(in the context of the process knowledge)
Next-Goal
(what to try to achieve next)
Performance Knowledge
(knowledge of how effective tasks are)
Add to
Select
Task
(what to do next and who should be responsible for it)
Do it —
(until termination condition satisfied)
Add to
New Process Knowledge
New Performance Knowledge

Fig. 1. Knowledge-driven process management (a simplified view)

driven process, the "next goal" is chosen by the process patron. This choice is made using general knowledge concerning the process—called the *process knowledge*. The process patron then chooses the tasks to achieve that next goal. This choice may be made using general knowledge about the effectiveness of tasks—called the *performance knowledge*. So in so far as the process goal gives direction to goal-driven—and task-driven—processes, the growing body of process knowledge gives direction to knowledge-driven processes. The management of knowledge-driven processes is considerably more complex than the other two classes of process, see Fig. 1. But, knowledge-driven processes are "not all bad"—they typically have goal-driven sub-processes.

Task-driven processes may be managed by a simple reactive agent architecture based on event-condition-action rules [5]. Goal-driven processes may be modelled as state and activity charts [11] and managed by plans that can accommodates failure [12]. Such a planning system may provide the deliberative reasoning mechanism in a BDI agent architecture [12] and is used in a goal-driven process management system [7] where tasks are represented as plans for goal-driven processes. But the success of execution of a plan for a goal-driven process is not necessarily related to the achievement of its goal. One reason for this is that an instance may make progress outside the process management system—two players could go for lunch for example. So each plan for a goal-driven process should terminate with a check of whether its goal has been achieved.

Managing knowledge-driven processes is rather more difficult, see Fig. 1. The role of the process knowledge is described in Sec. 3 and the role of the performance knowledge is described in Sec. 4.

3. Process knowledge and goals

Process knowledge is the wisdom that has been accumulated, particularly that which is relevant to the process instance at hand. For knowledge-driven processes the management of the process knowledge is shown on the left-hand side of Fig. 1. For knowledge-driven processes, management of the process knowledge is impractical.

The process knowledge in any real application includes an enormous amount of general and common sense knowledge. For example, the process trigger "the time is right to look at the US market" may be based on a large quantity of empirical knowledge and a fund of experiential knowledge. So the system does not attempt to represent the process knowledge in any way; it is seen to be largely in the heads of the users. The system does assist in the maintenance of the process knowledge by ensuring that any virtual documents generated during an activity in a knowledge-driven sub-process are passed to the process patron when the activity is complete. Virtual documents are either interactive web documents or workspaces in the LiveNet workspace system which is used to handle virtual meetings and discussions.

The system records, but does not attempt to understand the process goal. Any possible revisions the process goal are carried out by the patron without assistance from the system. Likewise the decomposition of the process goal to decide "what to do next"—the next-goal. It may appear that the system does not do very much at all! If the next-goal is the goal of a goal-driven process—which it may well be—then the system may be left to manage it as long as it has plans in its plan library to achieve that next-goal. If the system does not have plans to achieve such a goal then the user may be able to quickly assemble such a plan from existing components in the plan library. The organisation of the plan library is a free-form, hierarchic filing system designed completely by each user. Such a plan only specifies what has to be done at the host agent. If a plan sends something to another agent with a sub-goal attached it is up to that other agent to design a plan to deal with that sub-goal. If the next-goal is the goal of a knowledge-driven process then the procedure illustrated in Fig. 1 commences at the level of that goal.

So for this part of the procedure, the agent provides assistance with updating the process knowledge, and if a next-goal is the goal of a goal-driven sub-process then the system will manage that sub-process, perhaps after being given a plan to do so.

4. Performance knowledge

Performance knowledge is knowledge of how effective people, methods and plans are at achieving various things. For knowledge-driven processes the management of the performance knowledge is shown on the left-hand side of Fig. 1. Performance knowledge is substantially ignored by many workflow management systems. It is crucial to the efficient management of industry processes. The performance knowledge is used to support task selection—ie who does what—through inter-agent negotiation, see Sec. 5. So its role is a comparative one; it is not required to have absolute currency. With this use in mind, the *performance knowledge* comprises performance statistics on the operation of the system down to a fine grain of detail. These performance statistics are proffered by an agent in bids for work. To evaluate a bid, the receiving agent evaluates its meaning of payoff in terms of these statistics. If a parameter, p, can reasonably be assumed to be normally distributed, the estimate for the mean of p, μ_p, is revised on the basis of the i'th observation ob_i to $\mu_{p_{new}} = (1 - \alpha) \times ob_i + \alpha \times \mu_{p_{old}}$ which, given a starting value $\mu_{p_{initial}}$, and some constant α, $0 < \alpha < 1$, approximates the geometric mean $\frac{\sum_{i=1}^{n} \alpha^{i-1} \times ob_i}{\sum_{i=1}^{n} \alpha^{i-1}}$ where i = 1 is the most recent observation. In the same way, an estimate for $\sqrt{\frac{2}{\pi}}$ times the standard deviation of p, σ_p, is revised on the basis of the i'th observation ob_i to $\sigma_{p_{new}} = (1 - \alpha) \times | ob_i - \mu_{p_{old}} | + \alpha \times \sigma_{p_{old}}$ which, given a starting value $\sigma_{p_{initial}}$, and some constant α, $0 < \alpha < 1$, approximates the geometric mean $\frac{\sum_{i=1}^{n} \alpha^{i-1} \times | ob_i - \mu_p |}{\sum_{i=1}^{n} \alpha^{i-1}}$. The constant α is chosen on the basis of the stability of the observations. For example, if $\alpha = 0.85$ then "everything more than twenty trials ago" contributes less than 5% to the weighted mean; if $\alpha = 0.70$ then "everything more than ten trials ago" contributes less than 5% to the weighted mean, and if $\alpha = 0.50$ then "everything more than five trials ago" contributes less than 5% to the weighted mean.

Each individual agent/user pair maintains estimates for the three parameters: *time*, *cost* and *likelihood of success* for the execution of all of its plans, sub-plans and activities. "All things being equal" these three parameters are assumed to be normally distributed—the case when "all things are *not* equal" is considered below. *Time* is the total time taken to termination. *Cost* is the actual cost of the of resources allocated. The *likelihood of success* observations are binary—ie "success" or "fail"—so this parameter is binomially distributed, and is approximately normally distributed under the standard conditions.

Unfortunately, *value* is often very difficult to measure. For example in assessing the value of an appraisal for a bank loan, if the loan is granted then when it has matured its value may be measured, but if the loan is not granted then no conclusion may be drawn. The value of sub-processes are typically "less measurable" than this bank loan example. Although some progressive organisations employ experienced staff specifically to assess the value of the work of others. The existing system does not attempt to measure *value*; each individual represents the perceived *value* of each other individual's work as a constant for that individual.

Finally, the *allocate* parameter for each individual is the amount of work w_i^j, allocated to individual j in discrete time period i. In a similar way to *time* and *cost*, the mean *allocate* estimate for individual j is made using $\text{allocate}_{new} = (1 - \alpha) \times w^j + \alpha \times \text{allocate}_{old}$, where w^j is the most recent observation for individual j. In this formula the weighting factor α is chosen on the basis of the number of individuals in the system, and the relationships between the length of the discrete time interval and the expected length of time to deal with the work. The allocate parameter does not represent workload. It is not normally distributed and its standard deviation is not estimated. An estimate of *workload* is given by (allocations in) – (allocations out). The *allocate* and *value* estimates are associated with individuals. The *time*, *cost* and *likelihood of success* estimates are attached to plans.

The three parameters *time*, *cost* and *likelihood of success* are assumed to be normally distributed subject to "all things being equal". One virtue of the assumption of normality is that it provides a basis on which to query unexpected observations. Having made observation ob_{i+1} for parameter p, estimates for μ_p and σ_p are calculated. Then the next observation, ob_i, should lie in the confidence interval: $(\mu_p \pm \alpha \times \sigma_p)$ to some chosen degree of certainty. For example, this degree of certainty is 95% if $\alpha = 1.645$. The set of observations $\{ob_i\}$ can progressively change without individual observations lying outside this confidence interval; for example, an individual may be gradually getting better at doing things. But if an observation lies outside this confidence interval then there is grounds, to the chosen degree of certainty, to ask why it is outside.

Inferred explanations of *why* an observation is outside expected limits may sometimes be extracted from observing the interactions with the users and other agents involved. For example, if Person X is unexpectedly slow in attending to a certain process instance then a simple interchange with X's agent may reveal that Person X will be working on the company's annual report for the next six days; this may be one reason for the unexpected observation. Inferred knowledge such as this gives *one* *possible cause* for the observed behaviour; so such knowledge enables us to *refine*, but *not* to *replace*, the historical estimates of parameters.

The measurement ob_i may lie outside the confidence interval for four types of reason:

1) there has been a permanent change in the environment or in the process management system—the measurement ob_i is now the expected value for μ_p—in which case the estimates $\mu_{p_{old}}$ and $\sigma_{p_{old}}$ should be re-initialised.
2) there has been a temporary change in the environment or in the process management system and the measurements $\{ob_i\}$ are expected to be perturbed in some way for some time—in which case the reason, Γ, for this expected perturbation should be sought. For example, a new member of staff may have been delegated the responsibility—temporarily—for this sub-process. Or, for example, a database component of the system may be behaving erratically.
3) there has been no change in the environment or in the process management system and the unexpected measurement ob_i is due to some feature γ that distinguishes the nature of this sub-process instance from those instances that were used to calculate $\mu_{p_{old}}$ and $\sigma_{p_{old}}$. In other words, what was thought to be a single sub-process type is really two or more different—but possibly related—process types. In which case a new process is created and the estimates $\mu_{p_{old}}$ and $\sigma_{p_{old}}$ are initialised for that process.
4) there has been no change in the environment or in the process management system and the nature of the most recent process instance is no different from previous instances—the unexpected measurement ob_i is due to—possibly combined—fluctuations in the performance of individuals or other systems.

In option 2) above the reason Γ is sometimes inferred by the system itself. This has been achieved in cases when a user appears to be preoccupied working on another task. If the reason Γ is to be taken into account then some forecast of the future effect of Γ is required. If such a forecast effect can be quantified—perhaps by simply asking a user—then the perturbed values of $\{ob_i\}$ are corrected to $\{ob_i \mid \Gamma\}$ otherwise the perturbed values are ignored.

5. Agent Interaction

This section concerns the selection of a task for a given now-goal as shown in the middle of Fig. 1. The selection of a plan to achieve a next goal typically involves deciding *what* to do and selecting *who* to ask to assist in doing it. The selection of what to do and who to do it can not be subdivided because one person may be good and one form of task and bad at others. So the “what” and the “who” are considered together. The system provides assistance in making this decision. Sec. 4 describes how performance knowledge is attached to each plan and sub-plan. For plans that involve one individual only this is done for instantiated plans. That is there are estimates for each individual and plan pair. In this way the system offers advice on choosing between individual A doing X and individual B doing Y. For plans that involve more than one individual this is done by explicitly delegating the responsibility for populating that plan. So if a plan involves forming a committee then it is embedded in a plan that gives an individual the responsibility for forming that committee, and then estimates are gathered for the performance of the second of these.

There are two basic modes in which the selection of "who" to ask is done. First the *authoritarian* mode in which an individual is told to do something. Second the *negotiation* mode in which individuals are asked to express an interest in doing something. This second mode is implemented using contract nets with focussed addressing [13] with inter-agent communication being performed in KQML [14]. When contact net bids are received the successful bidder has to be identified. So no matter which mode is used, a decision has to be made as to whom to select. The use of a multiagent system to manage processes expands the range of feasible strategies for delegation from the authoritarian strategies described above to strategies based on negotiation between individuals. Negotiation-based strategies that involves negotiation for each process instance are not feasible in manual systems for every day tasks due to the cost of negotiation. If the agents in a multiagent system are responsible for this negotiation then the cost of negotiation is may be negligible.

If the agent making a bid to perform a task has a plan for achieving that task then its user may permit the agent to construct a bid automatically. As the bids consist of six meaningful quantities, the user may opt to construct a bid manually. A bid consists of the five pairs of real numbers (Constraint, Allocate, Success, Cost, Time). The pair *constraint* is an estimate of the earliest time that the individual could address the task—ie ignoring other non-urgent things to be done, and an estimate of the time that the individual would normally address the task if it "took its place in the in-tray". The pair Allocate is the mean of allocations-in and the mean of allocations-out. The pairs Success, Cost and Time are estimates of the mean and standard deviation of the corresponding parameters as described above. The receiving agent then:

- attaches a subjective view of the *value* of the bidding individual;
- assesses the extent to which a bid should be downgraded—or not considered at all—because it violates process constraints, and
- selects an acceptable bid, if any, possibly by applying its 'delegation strategy'.

If there are no acceptable bids then the receiving agent "thinks again".

6. Conclusion

Managing trans-corporate industry processes involves managing processes of three distinct types [5]. The management of knowledge-driven processes is not widely understood and has been described here. A multi-agent system manages goal-driven processes and supports the management of knowledge-driven processes [7]. The conceptual agent architecture is a three-layer BDI, hybrid architecture. During a process instance the responsibility for sub-processes may be delegated, and possibly out-sourced in an e-commerce environment. The system forms a view on who should be asked to do what at each step in a process, and tracks the resulting delegations of process responsibility. The system has been trialed on an emergent process application in a university administrative context.

References

[1] Robert Skinstad, R. "Business process integration through XML". In proceedings XML Europe 2000, Paris, 12-16 June 2000.

[2] Feldman, S. "Technology Trends and Drivers and a Vision of the Future of e-business." In proceedings 4th International Enterprise Distributed Object Computing Conference, September 25-28, 2000, Makuhari, Japan.

[3] Dourish, P. "Using Metalevel Techniques in a Flexible Toolkit for CSCW Applications." ACM Transactions on Computer-Human Interaction, Vol. 5, No. 2, June, 1998, pp. 109—155.

[4] A. P. Sheth, D. Georgakopoulos, S. Joosten, M. Rusinkiewicz, W. Scacchi, J. C. Wileden, and A. L. Wolf. "Report from the NSF workshop on workflow and process automation in information systems." SIGMOD Record, 25(4):55—67, December 1996.

[5] Debenham, J.K. "Three Intelligent Architectures for Business Process Management", in proceedings 12th International Conference on Software Engineering and Knowledge Engineering SEKE2000, Chicago, 6-8 July 2000.

[6] Jain, A.K., Aparicio, M. and Singh, M.P. "Agents for Process Coherence in Virtual Enterprises" in Communications of the ACM, Volume 42, No 3, March 1999, pp62—69.

[7] Debenham, J.K. "Supporting knowledge-driven processes in a multiagent process management system." In proceedings Twentieth International Conference on Knowledge-Based Systems and Applied Artificial Intelligence, ES'2000: Research and Development in Intelligent Systems XV, Cambridge UK, December 2000.

[8] Fischer, L. (Ed) "Workflow Handbook 2001." Future Strategies, 2000.

[9] Debenham, J.K. "Supporting Strategic Process", in proceedings Fifth International Conference on The Practical Application of Intelligent Agents and Multi-Agents PAAM2000, Manchester UK, April 2000.

[10] Debenham, J.K. "Knowledge Engineering: Unifying Knowledge Base and Database Design", Springer-Verlag, 1998

[11] Muth, P., Wodtke, D., Weissenfels, J., Kotz D.A. and Weikum, G. "From Centralized Workflow Specification to Distributed Workflow Execution." In Journal of Intelligent Information Systems (JIIS), Kluwer Academic Publishers, Vol. 10, No. 2, 1998.

[12] Rao, A.S. and Georgeff, M.P. "BDI Agents: From Theory to Practice", in proceedings First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, USA, pp 312—319.

[13] Durfee, E.H.. "Distributed Problem Solving and Planning" in Weiss, G. (ed). Multi-Agent Systems. The MIT Press: Cambridge, MA.

[14] Finin, F. Labrou, Y., and Mayfield, J. "KQML as an agent communication language." In Jeff Bradshaw (Ed.) Software Agents. MIT Press (1997).