2009 Fourth IEEE International Conference on Global Software Engineering

TAMRI: A Tool for Supporting Task Distribution in Global Software Development Projects

Ansgar Lamersdorf
University of Kaiserslautern
a_lamers@informatik.uni-kl.de

Jürgen Münch
Fraunhofer IESE
juergen.muench @iese.fraunhofer.de

Abstract

The distribution of tasks to sites is one central activity in global software development project planning. Due to the large number of assignment possibilities, tool support seems to be adequate for supporting the evaluation and selection of task assignments. We present TAMRI, a planning tool for identifying task assignments based on multiple criteria and weighted project goals. Its implementation combines a distributed systems approach with Bayesian networks. The tool can be adapted to specific organizational environments by exchanging the underlying Bayesian network. The article presents an overview of task distribution approaches, gives three application scenarios for the tool, and shows the implementation of the tool as well as its application in the scenarios.

1. Introduction

Besides the remarkable success and widespread distribution of Global Software Development (GSD) [3], many problems in GSD still remain unsolved, such as lack of informal communication [8], lack of trust [13], time differences [5], or cultural differences [9]. Accordingly, the failure rate in GSD projects is remarkably high [6].

As the challenges and problems of distributed development cover various different aspects and phases of a development project, they are expected to have a significant impact on the project management level, and project planning should proactively find ways to cope with the inherent specifics of GSD [12].

One central aspect in GSD project planning is the assignment of development tasks to available sites: A software development process consists of different tasks (e.g., design, implementation of different components) requiring different capabilities. In GSD, these tasks usually have to be distributed and allocated to sites. This influences both the benefits and the risks of distributed development. Different strategies can be implemented here, such as "nearshoring" to reduce problems of cultural and time-zone differences [4] or "follow-the-sun" to reduce development time [14].

However, many different goals and criteria have to be regarded simultaneously in task allocation, for instance cultural and time-zone differences, capabilities at the sites, and labor cost structures. In addition, there exists a potentially very large number of possible allocations. For example, if a software development project consists of 10 independent development tasks that can be distributed across 3 development sites, theoretically $3^{10} = 59049$ different combinations of allocating tasks to sites exist. Even if just a fraction of these combinations is possible in practice due to organizational or logistic restrictions, overseeing and evaluating all possibilities with respect to multiple criteria can be hard or impossible to do for a human in a real situation.

We developed TAMRI (Task Allocation based on Multiple cRIteria), a tool that can support project managers in identifying suitable task allocations in a GSD project planning process. The tool implements an algorithm that uses (1) a set of weighted goals, (2) characteristics of the project tasks and the available sites, and (3) causal relationships between characteristics and goals for deriving a weighted list of suggestions for task allocations. As distributed software development processes and their characteristics can differ widely between organizations, the tool is generic, making it possible to adapt goals, characteristics, and causal relationships to specific environments.

The remainder of this article is structured as follows: First, a brief overview of approaches and algorithms for task distribution will be given in order to motivate the selection of the implemented algorithms. Then, application scenarios for using the tools will be presented, followed by an overview of the tool's implementation. Section five will show the application of the tool in the given scenarios. Limitations and a discussion of the approach will conclude the article.

978-0-7695-3710-8/09 $25.00 © 2009 IEEE
DOI 10.1109/ICGSE.2009.50

2. Task Distribution Approaches

In a previous study done by the authors [10], several approaches for distributing tasks over nodes in a network were analyzed in terms of their applicability as a task distribution model. For this purpose, a list of requirements for a distribution model was derived. These included (1) consideration of multiple, perhaps conflicting objectives (e.g., time, costs), (2) the ability to describe both characteristics of and dependencies between tasks and nodes, (3) adaptability (e.g., for including new objectives or characteristics), (4) formality, and (5) an empirical basis for the underlying causal relations.

Approaches from three domains were analyzed: global software development, distributed production, and distributed systems. Table 1 gives an overview of the result of the evaluation. The detailed evaluation and the references to the approaches can be found in [10].

Table 1. Comparison of distribution models [10] and requirement fulfilment (not -, partly o, mostly +, total ++), selected approach in bold print

Domain	Approach	Multi-objective goal	Properties		Dependencies		Adaptability	Formality	Empirically based
			Tasks	Nodes	Tasks	Nodes			
Distributed SW Development	Modification Requests	-	o	o	-	+	-	+	o
	Global Studio Project		+	+	+	-	-	-	+
	Distributed CoCoMo	-	+	++	-	-	o	+	o
	Simulation Model	+	-	+	o	+	o	+	o
	Reference Model for GSD	+	+	+	-	+	+	-	o
Distributed Production	Linear Programming	-	+	o	-	o	-	++	
	Plant Distribution	-	+	+	-	-	-	++	
	BMW Production	-	+	+	-	o	-	++	
Distributed Systems	**Processor Allocation**	**o**	**+**	**+**	**o**	**+**	**o**	**+**	
	Opportunity Costs	++	+	+	-	-	o	+	
	File Allocation	+	+	+	-	+	-	++	

The comparison showed that task distribution approaches from GSD often lack formality and consider only single aspects in task distribution. Distributed production approaches usually are very formal but lack multi-objectivity and adaptability. In the end, one approach for processor allocation [2] fulfilled most of the criteria (all except "empirical basis" were at least partially fulfilled). It can identify the optimal distribution of computing tasks over processors in a network based on quantitative descriptions of execution costs at and communication costs between the processor nodes.

As a result of the evaluation, it was decided to use this approach for the model underlying the project planning tool. However, several adaptations had to be made in order to reuse the algorithm for project planning in GSD [11]. For example, the original algorithm worked with quantitative, distinct cost functions (e.g., execution costs of task x at node y) without providing a method for deriving the values. We used Bayesian networks [1] for modeling the costs since this made it possible to a) describe causal relationships (e.g., the impact of time zone differences on development time) and to b) reflect the uncertainties immanent in describing human behavior.

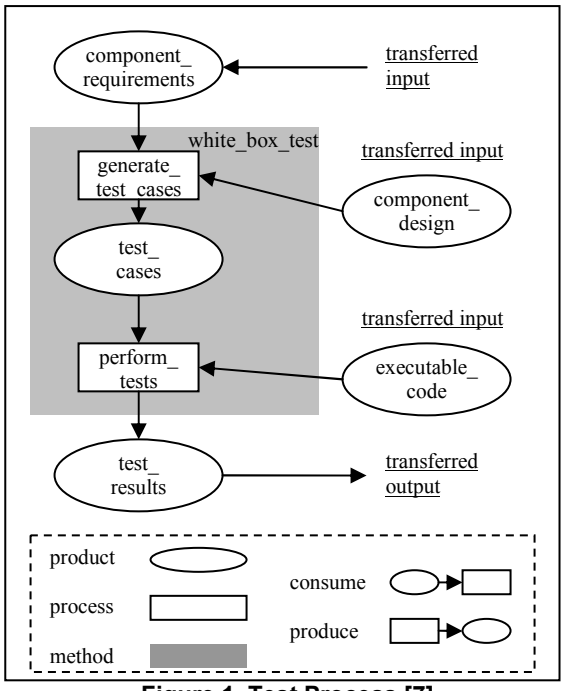


Figure 1. Test Process [7]

3. Application Scenario

The application scenarios are based on a detailed scenario for planning a distributed project that was defined by Goldmann et al. [7] as part of the MILOS

project. The MILOS scenario focuses on a distributed planning process for component development and testing. Two types of activities are to be distributed in this scenario. Their specifications are shown in Figures 1 and 2.

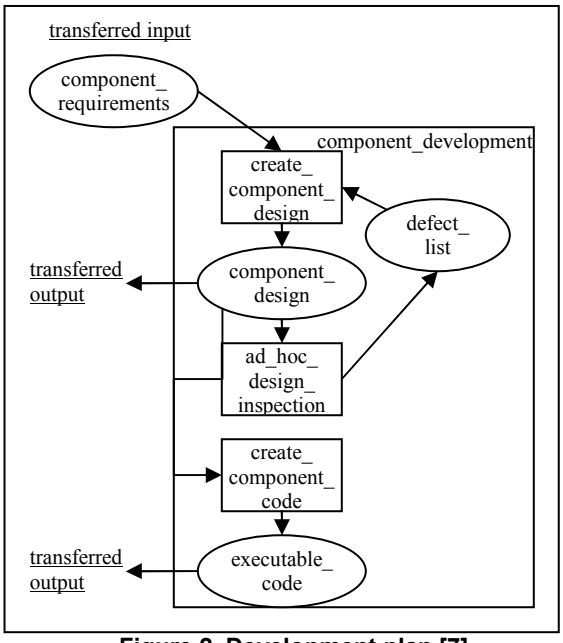


Figure 2. Development plan [7]

In the MILOS scenario, a development and a test site are already selected and the planning begins with the assignment of individuals to tasks and the selection of concrete techniques for tasks. We extend this scenario by adding the step of choosing a development or testing site and define three new scenarios. In our new scenarios, three sites (A, B, C) are available. Figure 3 shows some characteristics of the sites.

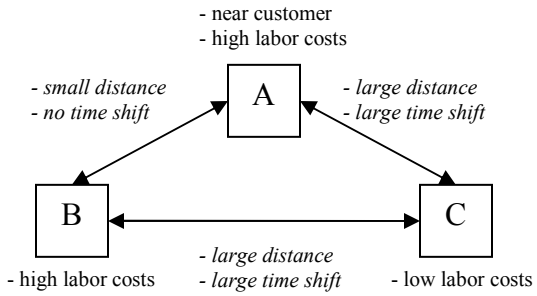


Figure 3. Available Sites

In the following, the three new scenarios are described in such a way that they can serve as use cases for the TAMRI tool:

Scenario 1: One component is to be developed and tested. It was already decided to assign the development to site A. Testing can now be assigned either to site B or C. As component development and testing are complementary activities, round-the-clock development can be applied in principle. The time shift between A and C would make round-the-clock development possible if testing was assigned to C. However, at B, people are more familiar with the component and thus have higher testing skills. Cost rates are not considered.

Scenario 2: Three components are to be developed and tested. The components are coupled differently, thus requiring different amounts of communication between the teams developing them. Figure 4 gives an overview of the tasks. From the viewpoint of expertise and skills, the best abilities are located at A for developing the components and at B for testing them. However, labor costs are much lower at C and the time shift to C would enable round-the-clock development.

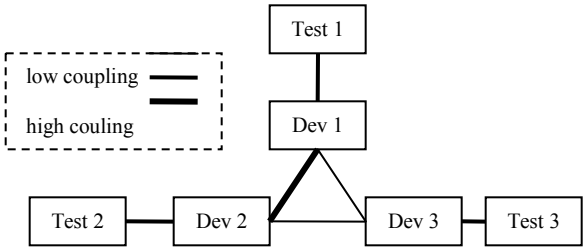


Figure 4. Tasks in scenario 2

Scenario 3: Now, the development step (Figure 2) is split into design and (code) development. Three components are to be designed, developed, and integrated into the final product (see Figure 5). Integration has to be done at the customer site A. Design would also benefit from being at the customer site, but the people at C have the most expertise in design. Due to available staff, development can only be done at B or C.

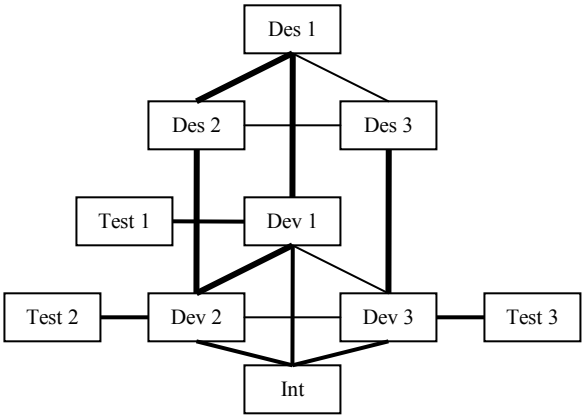


Figure 5. Tasks in scenario 3

4. Tool Implementation

The TAMRI tool was implemented in Java and will be presented in the following.

4.1. Implemented Algorithm

The distributed system algorithm selected before was used for identifying distributions of tasks over sites. As input, this algorithm uses numeric values for both the cost of executing a task at a site and the communication costs between tasks located at different sites. Thus, a way had to be developed for modeling these costs. This was done using the Bayesian network (BN) approach [1].

Figure 6 gives an example excerpt of a BN describing the communication costs between two tasks located at different sites. It shows the impact of task and site characteristics on development time. For example, a time shift between sites increases communication problems between sites, which, in turn increases the overhead for two coupled tasks assigned to these sites and increases development time. On the other hand, a time shift can increase the potential benefit of round-the-clock development, which may decrease development time. Using BNs, it is possible to describe causal relationships between various variables and project goals.

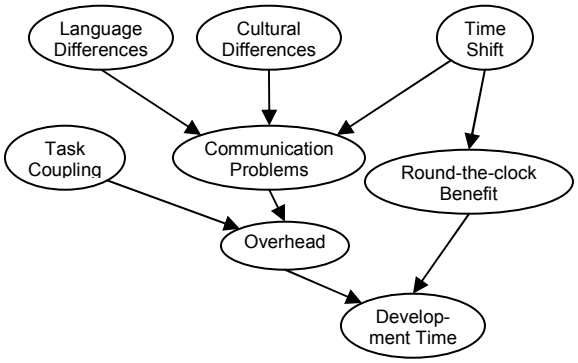


Figure 6. BN excerpt for communication cost

As a result, the BNs produce statistical distributions over the possible communication and execution costs (e.g., "Task t assigned to site s would result in execution costs x with probability p and y with probability q"). However, the distributed systems algorithm requires numeric values as input (e.g., "t assigned to s always results in execution costs x"). This discrepancy was handled by using stochastic simulation: The distributed systems algorithm is executed n times. In every run, the cost values are set randomly according to the distributions produced by the BNs (e.g., the cost for executing t at s is set to x with probability p and to y with probability q). As one "optimal" distribution is produced for every run, the algorithm is able to return a weighted list of suggestions (e.g., "distribution d_1 is optimal with probability p_1 and d_2 is optimal with probability p_2").

The overall algorithm is summarized in Figure 7. As shown in the overview, the different goals are integrated into one cost value for each BN by normalizing, weighting, and adding them. Thus, it is possible to weight the goals according to project-specifics.

For every task and every site:
- Instantiate a BN for execution costs based on project and resources characteristics

For every combination of two tasks and two sites:
- Instantiate a BN for communication costs

For n times (n >> 1, e.g., n = 2000):
- Select random values for every goal in every BN according to the probability distributions
- Integrate the different goal values into one cost value per BN (normalizing, weighting, and adding)
- Execute the distributed system algorithm with the execution cost and transmission cost values from the previous step
- Add the returned assignment to the list of optimal assignments

Return the list of assignments in decreasing order of occurrences

Figure 7. Algorithm overview

4.2. Adaptability

In this implementation, the goals, influencing variables, and their causal relationships are stored solely in the Bayesian networks. In the prototypical version presented in this article, they stem from a combined literature and interview study [11]. However, in specific organizations, the underlying relationships may be different. For example, other goals and variables might be relevant, or the impact of a certain variable on project goals might be different. This would result in individual BNs covering specific organizational environments

We addressed this by making the implementation of the tool adaptable in the following way: The model implementation consists of a generic part, containing the algorithm and a framework for Bayesian networks. An organization-specific part contains the description of a concrete Bayesian network. Accordingly, the user interface consists of a generic part and a specific implementation that describes the masks for setting the individual parameters.

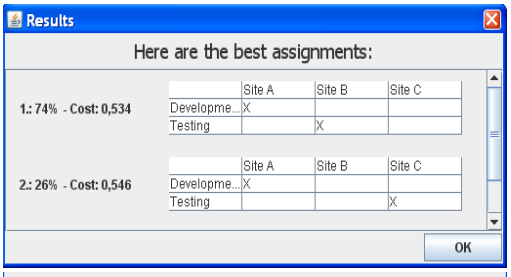
With this architecture, it was possible to implement a new organization-specific model (i.e., the Bayesian networks and the user interface) in less than two days.

5. Tool Application for the Scenarios

In the following, the use of the tool will be demonstrated for the scenarios stated in Section 3 with the project goals cost, quality, and development time.

Scenario 1: Figure 8 shows the results for scenario 1. If the strongest weight is put on quality (a), the tool suggests assigning testing to B. However, if more emphasis is put on development time (b), the model slightly favors C as testing site due to the possibility of round-the-clock development.

(a)



(b)

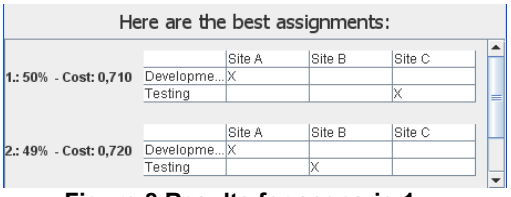


Figure 8.Results for scenario 1

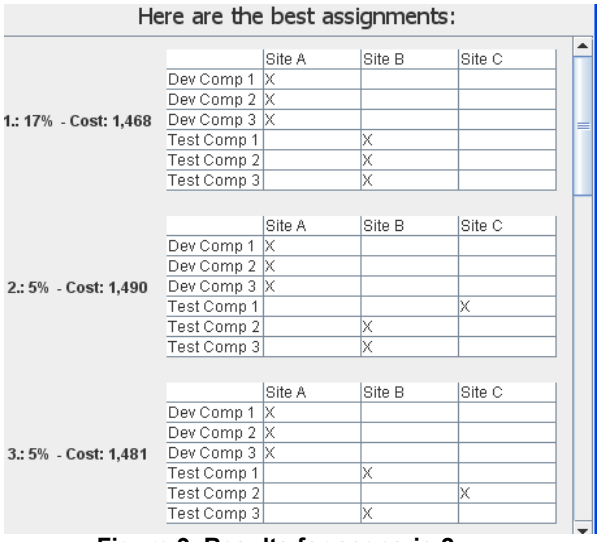
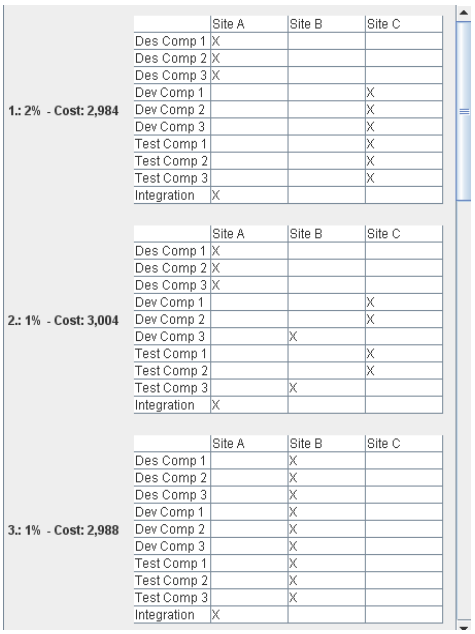


Figure 9. Results for scenario 2

Scenario 2: Figure 9 shows the result for scenario 2. It shows that, due to the emphasis on quality, development should be done at A and testing at B. However, single test tasks could also be done at C in order to profit from the low labor rates and the possibility of round-the-clock development.

Scenario 3: Figure 10 shows the result for scenario 3 with an emphasis on quality (a) and cost (b). It can be seen that due to the strong coupling between the different tasks, it is suggested to keep most tasks together at one site. If more emphasis is put on cost, then the tool suggests assigning large chunks of work to C, as it is the site with the lowest labor rates.

(a)



(b)

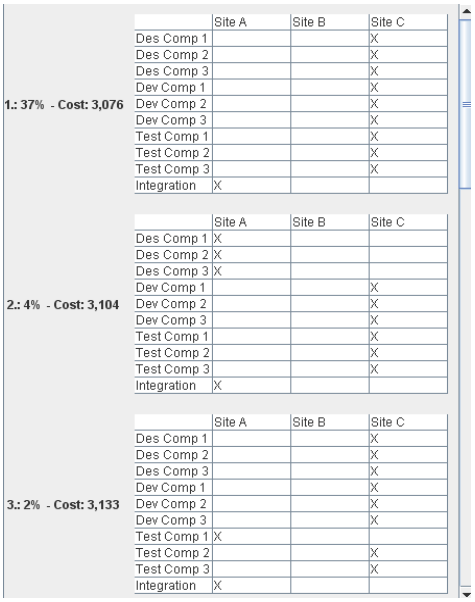


Figure 10. Results for scenario 3

6. Limitations and Discussion

To our best knowledge, there exists no similar tool for supporting task allocation decisions in global software development. Therefore, our work aims at better

understanding and supporting the development of an improved and more systematic process of project planning and task distribution in GSD. However, there are some limitations and open questions that can impede the application of this or similar tools in practice:

The application of the tool requires a very large amount of knowledge on causal relationships in distributed development. Few causal relationships can be gathered from results of empirical studies or theoretical analyses; most of the knowledge has to be gathered individually at each organization. It is doubtable that many companies possess the maturity and data basis needed to gather this knowledge. Therefore, the Bayesian networks would probably have to be developed based on expert estimations and not on empirical data, which would increase the uncertainty.

The same is expected for determining and quantifying the characteristics of the resources (e.g., cultural differences between two sites) or of the project (e.g., coupling between two tasks). Most factors will be described with a large amount of uncertainty and others might not be known at all. Bayesian networks and the tool allow not setting all parameters to known values, but the less information is known on the project, the more uncertainty will be in the result.

Another limitation of the tool is the fact that it uses a "black-box" algorithm: Due to the large number of calculations executed by the tool, it is not transparent to the user why a certain distribution is suggested by the algorithm. It is left to the user to conclude the rationale for the suggestion. To many managers, it might be unacceptable to follow the suggestions of a tool without being able to reconstruct the underlying reasons in detail.

However, despite these limitations that might constitute obstacles to usage in an industrial context, we see the TAMRI tool as a good starting point for exploring the possibilities of tool-supported project planning and task allocation in global software development.

In our future work, we aim at overcoming the limitations of the tool. Therefore, we will look at (1) techniques for modeling distributed development without needing access to large amounts of empirical data and (2) algorithms or processes that can support task allocation decisions in a more interactive way. In addition, we will try to build empirically-based models for describing the impact of distributed development on project goals, in order to use them for future decision support tools.

References

[1] Ben-Gal, I.: Bayesian Networks. In Ruggeri, F., Kenett, R., Faltin, F. (editors): *Encyclopedia of Statistics in Quality and Reliability*, John Wiley & Sons (2007).

[2] Bokhari, S. H.: A Shortest Tree Algorithm for Optimal Assignments across Space and Time in a distributed Processor System. *IEEE Transactions on Software Engineering* SE-7:6, pp. 583-589 (1981)

[3] Damian, D., Moitra, D.: Global Software Development: How Far Have We Come? *IEEE Software*, 23(5): pp. 17-19 (2006)

[4] Dubie, D.: Outsourcing Moves Closer to Home. *CIO Today*, December 18, 2007

[5] Espinosa, J. A., Carmel, E.: Modeling Coordination Costs Due to Time Separation in Global Software Teams. *International Workshop on Global Software Development*, 2003

[6] Fabriek, M., Brand, M. van de, Brinkkemper, S., Harmsen, F., Helms, R.W.,: Reasons for Success and Failure in Offshore Software Development Projects. *European Conference on Information Systems*, 2008

[7] Goldmann, S., Münch, J., Holz, H.: Distributed Process Planning Support with MILOS. *International Journal of Software Engineering and Knowledge Engineering*, vol. 10, no. 4, pp. 511-525, October 2000

[8] Herbsleb, J.D., Mockus, A.: An Empirical Study of Speed and Communication in Globally-Distributed Software Development. *IEEE Transactions on Software Engingeering*, vol. 29, no. 6, June 2003

[9] Krishna, S., Sahay, S., Walsham, G.: *Managing cross-cultural issues in Global Software Outsourcing*. Communications of the ACM 47, 4 (Apr. 2004), pp. 62–66

[10] Lamersdorf, A., Muench, J., Rombach, D.: Towards a Multi-Criteria Development Distribution Model: An Analysis of Existing Task Distribution Approaches. *International Conference on Global Software Development*, ICGSE 2008

[11] Lamersdorf, A., Muench, J., Rombach, D.: A Decision Model for Supporting Task Allocation Processes in Global Software Development. *International Conference on Product Focused Software Development and Process Improvement* PROFES 2009, in press

[12] Lee, G., DeLone, W., Espinosa, J. A.: Ambidextrous coping strategies in globally distributed software development projects. *Communications of the ACM*, Volume 49, Issue 10 (October 2006)

[13] Smite, D., Moe, N. B.: Understanding a Lack of Trust in Global Software Teams: A Multiple-Case Study. *International Conference on Product Focused Software Development and Process Improvement* PROFES 2007: 20-34

[14] Treinen, J. J., Miller-Frost, S. L.: Following the Sun: Case studies in global software development. *IBM Systems Journal*; Oct-Dec 2006; 45, 4, p. 773