Fragments for Composing Collaborative Systems

I.T. Hawryszkiewycz, B. Henderson-Sellers and N. Tran

Faculty of Information Technology
University of Technology Sydney
Australia
igorh@it.uts.edu.au

Abstract: Collaborative systems are characterized by groups or communities jointly working to accomplish some organizational goal. Members of the communities must coordinate their efforts to ensure that such goals are met. Development of collaborative systems can be facilitated by identifying common aspects of such coordination and by constructing product method fragments that can be composed into working systems. This paper uses a metamodel to provide a basis for such fragments, describing the metamodel and defining typical fragments.

1 Introduction

Collaboration is now becoming an important component of most business processes especially those where knowledge work is important. However, few methodologies provide direct support for integrating collaboration into business systems. One reason for this may be that collaboration evolves as a business process situation changes and such evolution is difficult to support in most methodologies. A second reason may be the need to provide a variety of options to support business process collaborators. Still a third is an absence of any collaborative policies within most organizations and hence little emphasis on supporting collaboration in applications. These options must support different communication styles and consequently provide services to support such styles. This is usually prohibitive from a cost viewpoint and hence collaboration takes place outside the computer-based information system. This paper proposes a way of integrating collaboration into business processes by providing process and product fragments [RP96] that can be easily integrated into pre-existing methodologies.

A collaborative system is defined as one where activities are jointly carried out by many participants. An alternative and more formal definition is where the emphasis is on closely coordinated teams to carry out the process. Designing collaborative groupware systems must emphasize social concepts such as groups of people and the way their interaction evolves and the way they jointly contribute to system activities.

It is also the case that collaboration in different domains often has many similar activities. Team formation and coordination have common characteristics irrespective of whether they occur in software development or in product innovation. There is therefore

considerable opportunity for defining method fragments that can be reused in many applications.

Our goal is to identify fragments that represent generic concepts providing the basis for future reuse, especially for object-oriented and agent-oriented systems. This can then make policies for collaboration more feasible. Most contemporary conceptual modelling methods concern relatively stable processes and emphasize functions and processes rather then people. In stable processes, the flow of transactions can be predefined and the data structures are stable. The paper uses a metamodel of collaboration that defines such generic concepts as its basis and then uses it to define a set of concepts to precisely specify collaborative work and generic agents that correspond to these concepts.

2 Fragments

Fragments have been defined [HBO94, Br96] as coherent parts of a system engineering methodology, generally stored in some kind of repository a.k.a. methodbase (e.g. [Sa93, Br96]. They are an integral part of a method engineering approach [KW92] in which system development methods can be composed of fragments in such a way that the methodology is situational (e.g. [Br96]). A fragment usually supports some well-defined part of a methodology. This may be a process used to develop systems or a product used, created or modified by that process. It may also be a part that makes up the developed business system.

Fragments fall into two classes: process fragments and product fragments [RP96, Br96]. Together, these fragments can be combined to make up a new system development methodology (e.g. [He04]).

2.1 Defining product fragments for CSCW systems

This paper concentrates on product fragments, which can be used to build collaborative business applications, It uses the approach shown in Figure 1 in which a metamodel for collaborations is used to define an ontology for creating conceptual models for collaborative processes. The metamodel is expressed in terms of concepts. These concepts are then used to build a conceptual model of collaborative business applications.

Collaborative work is often supported by collaborative technologies. A frequently used step in system development is to directly map the metamodel concepts to groupware workspaces. Other steps include the conversion of the semantic model to object diagrams, which are then converted to workspace implementations. This paper concentrates on the metamodel aspects and how they can be used to identify usable fragments.

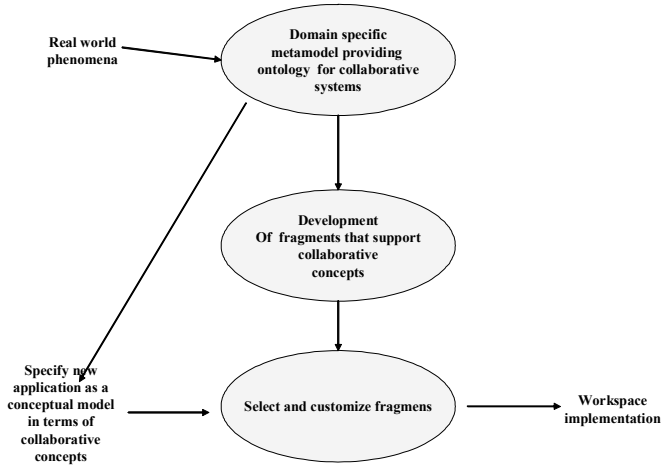

Figure 1 Creation and selection of method fragments for CSCW

3 A Metamodel for Collaborative Systems

The metamodel has been described in detail earlier [Ha05a] and has evolved and been verified through a variety of applications that include business networking [Ha96] and strategic planning [Ha97]. Organizational computational theory [CG99] provides a further foundation for the metamodel since metamodel concepts provide a basis for managing knowledge about collaborative relationships. Thus, rather than identifying data objects and processes, we identify organizational entities and their agencies.

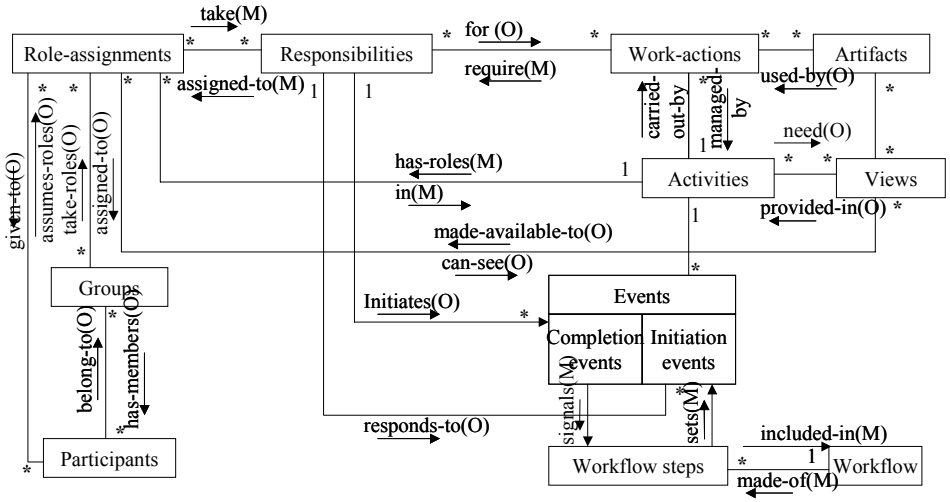

Figure 2 – Metamodel showing Collaborative Concepts

This also supports the notions of [RT00] of organizations supported by interacting agencies. The metamodel also provides a way to build agent-based systems that can provide active support to users LHH05].

3.1 A Metamodel for Collaborations

The major components in the proposed metamodel (Figure 2) are the group role and participant.

The concepts can be described in a simple way as:

concept <name>, '<description>':
{ {attributes: <attribute-name>}, {concept-knowledge},
{methods: <method-name>: {permissions:<permission, role-name>}},
{constraints},
};
relationship <name>,'<description'>:
{{ source-name:<concept>, target-name: <concept>}
{cardinality},
{attributes: <attribute-name>},
{methods: <method-name>: {permissions:<permission, role-name>}},
{matching-rules: <how relationship satisfies the needs of both concepts},
{communication: <message-name>},
{relationship-knowledge}
{constraints},
};

Here, concept knowledge concentrates on ways to accomplish the concept goal. Relationship knowledge, on the other hand, concentrates on the best ways to set up relationships between pairs of concepts. Some examples of the general approach now follow. These are indicative only, by illustrating the kinds of rules that make up the concepts and relationship definitions. Our goal now is to define fragments based on this metamodel.

4 Guidelines for selecting fragments

Our goal is to define fragments that are based on the metamodel concepts. Standard method engineering using a metamodel states that each fragment is, initially, an instance of one concept (one class in the metamodel). Some authors (e.g. [RP96, Ra04]) create chunks from pairs of fragments in that a work unit and a work product are inextricably coupled. Here, we avoid such coupling in order to retain flexibility but suggest the use of composite objects in the sense of recommended "patterns" of several fragments, conjoined to serve specific CSCW situations.

4.1 Composite objects

Composite objects have been defined earlier [SLW04]. Primarily, a composite object is one made up of a fixed structure of more elementary objects. As an example, we illustrate two such composite objects: Role Specification and Group Specification. The idea is illustrated in Figure 3. It shows the construction of two fragments. Each fragment can be used independently but can also be combined into a larger fragment. The group specification fragment is used to construct teams and can be used in any application. The interface is the role's responsibilities that can be undertaken by the different team members. The second fragment is focused on the organization of work actions into activities. The interface now represents the responsibilities that are needed to carry out the work actions. The two fragments can now be linked using the role responsibilities.

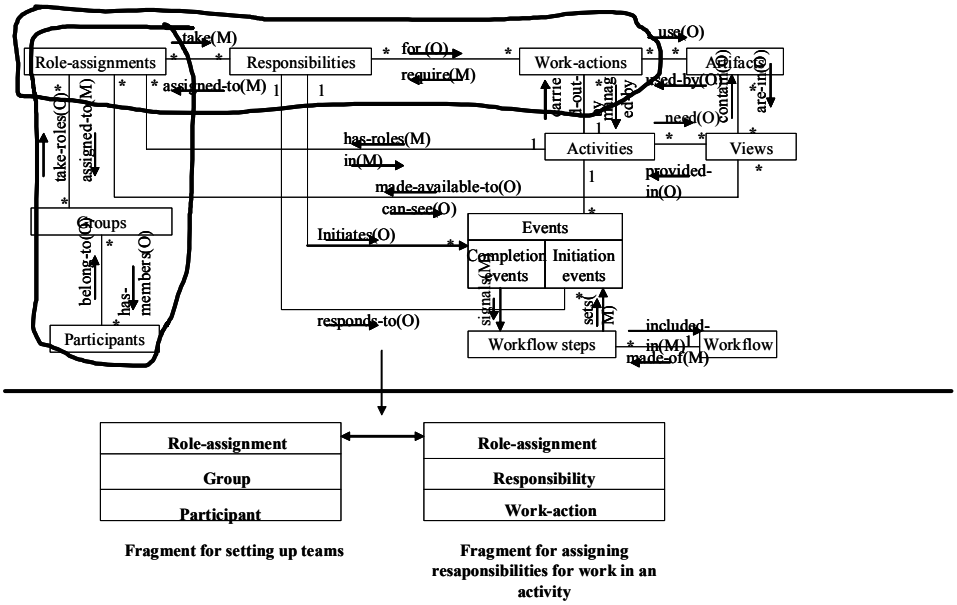

Figure 3 – Constructing fragments for collaborative work

Figure 3 illustrates two possible fragments. One of these is the Role Specification fragment where responsibilities are distributed for a particular activity. For example these may be responsibilities for a team completing a software task. They may also be responsibilities within a teaching institution. The other object is to specify a group with common interests usually concerned with sharing common knowledge or keeping track of some activities. This may also be a structured team that can be moved from one project to another, as for example can occur in the construction industry. There are a number of other possibilities for generic composite objects in collaborative environments. For example an e-portfolio fragment can combine views, artifacts and work-actions to provide a basis for joint management of documents.

A further alternative is to create a generic concept that can combine a number of more elementary fragment into a composite fragment. One such composite fragment, not shown here, is an engagement [Ha05b]. This includes the concepts found in group formation and work assignment but is more difficult to customize.

4.2 Role Specification

As shown in Figure 2, a collaborative system involves "groups" of "participants" taking "responsibilities" to carry out "work-actions" through their "role-assignments". Accordingly, a role-assignment entails a collection of responsibilities and a set of work actions to fulfil these responsibilities.

These three concepts (role-assignment, responsibility and work-action) are thus highly related, and their conjunction can be used to define any specific organizational role in a collaborative system. We thus propose a fragment (or "pattern") called "*Role Specification*", which aims to support the modelling of roles in collaborative organizations.

This fragment is a composite object made up of three component concepts, "role-assignment", "responsibility" and "work-action" (Figure 4). For example, in a collaborative system that represents a conference review board, two required role specifications are "Reviewer" and "Author". The "Reviewer" role specification will define a Reviewer role-assignment, which has the responsibility of reviewing papers and the work-actions of reading papers and providing comments on papers.

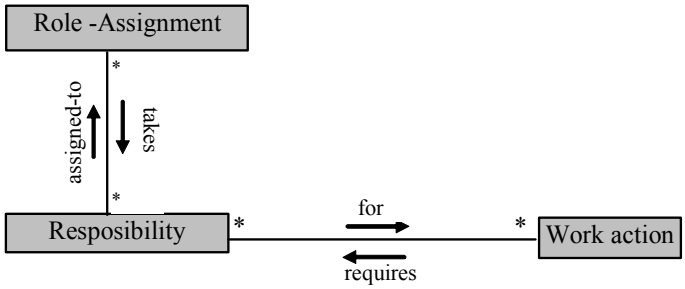

Figure 4 – Composite structure of Role Specification fragment No. This is a collection of elements in the metamodel. A fragment is, by definition, instantiated from the metamodel.

More formal definitions of some of the component concepts and relationships composing the Role Specification fragment are as follows.

concept: role-assignment:
{{attributes: (date-created: date)}
{methods: create-assignment, delete-assignment}

concept responsibilities;
{{attributes: (capability:<artifact>}
{concept-knowledge: context needed by responsibility, skills needed for responsibility}
{method: assemble-responsibility-skills}}

concept work-action;{{attribute: {work-kind, required resources, required knowledge, required resources}
{concept-knowledge: responsibilities-needed-given-assignment, expected-user-behaviour, optimum-monitoring}
{method: identify responsibilities, structure responsibilities}}

relationship require;
{{source-concept: < work-action >, target-concept: < responsibility >}.
{knowledge: best ways to structure responsibility, ways to set responsibility goals},
{matching-rule: 'work responsibility is needed to meet goal'},
{communication: assign-responsibility}.}

The Role Specification fragment will now combine all the features into the one fragment. In this case:

Fragment action-support:
{{attributes: (capability:<artifact>}
{knowledge: best ways to structure responsibility, ways to set responsibility goals, responsibilities-needed-given-assignment, expected-user-behaviour, optimum-monitoring}
{method: identify responsibilities, structure responsibilities, add-responsibility-to-role, delete-responsibility, add-responsibility, delete-responsibility, method: assemble-responsibility-skill}

Table 1 illustrates some example instantiations of the Role Specification fragment for a collaborative learning system. The system's goal is to support collaborative learning activities of teachers and students in an academic course. Some roles in Table 1 are *domain-specific*, that is, they relate specifically to the problem domain of the application, for example, "Teacher" and "Student" roles.

Some other roles are *group-specific*, that is, they deal particularly with the management and operation of teamwork, irrespective of the application domain; for instance, "Activity Manager" and "Artifact Manager" roles. Nevertheless, the developers do not have to define the domain-specific roles and group-specific roles separately.

They can, instead, embrace these two types of roles into an *application-specific role*. For example, "Lecturer-in-Charge" role is an application-specific role being made up of a domain-specific role "Teacher" and several group-specific roles, "Activity Manager"

and “Artifact Manager”. Any participant playing the role “Lecturer-in-Charge” will be assumed to be a teacher, a group manager, an activity manager *and* an artifact manager.

Example instances of Role Specification fragment	Role-assignment	Responsibility	Work-action
Teacher role specification	Teacher	Setting and assessing assignments	Create assignment Distribute assignment Collect Assignment Assess assignment
		Provide materials	Create learning materials Receive feedback
Student role specification	Student	Submit assignment	Read assignment questionnaire Produce assignment answers
		Study course materials	Read learning material Provide feedback
Activity Manager role specification	Activity Manager role	Manage/monitor a group’s activity (For example, a discussion)	Initiate discussion activity Monitor interactions during discussion Resolve conflicts during discussion Reinforce focus on discussion when off track Inform participating parties of the status/progress of discussion.
Artifact Manager role specification	Artifact Manager	Control access to group’s artifact (For example, a class’ noticeboard)	Set/change permissions to the noticeboard Grant appropriate permissions to requesting members Control members’ access to noticeboard
		Support artifact’s awareness	Inform new members of the noticeboard and any changes
Lecturer-In-Charge role specification	Lecturer-in-charge	Combination of responsibilities of the above roles.	Combination of work actions of Teacher role, Group Manager role, Activity Manager role and Artifact Manager role.

Table 1 – Example instances of Role Specification fragment for a collaborative learning system

A similar kind of fragment could be customized for software development. The teacher would be the team leader and students would be team members. Further specializations would also be possible, for example, a module tester. The activity manager could be the leader of a number of teams all involved in the same project.

4.3 Group Specification

According to the metamodel for collaborations (Figure 2), a "group" (or team) is composed of "participants" taking on role-assignments that determine their responsibilities and plausible work-actions. Since role-assignments, associated responsibilities and work-actions have been defined via the Role Specification fragment (section 4.2), we can define a group via a fragment "*Group Specification*", which is a composite object consisting of three concepts: "group", "participant" and "role specification" (Figure 5). A Group Specification serves as a template for modelling any specific group in a collaborative system. The "sub-group" relationship attached to the Group object in the Group Specification fragment (Figure 4) allows for the specification of parent-child association between groups.

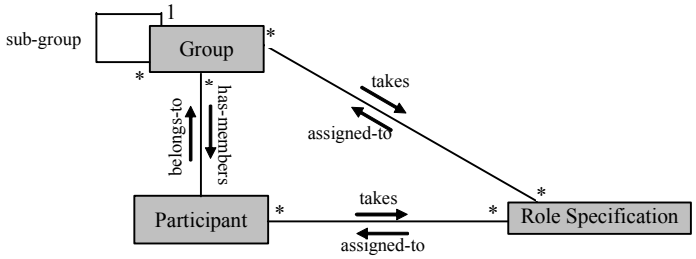

Figure 5 – Composite structure of Group Specification fragment

More formal definitions of some component concepts and relationships composing the Group Specification fragment are as follows.

relationship: has-members, 'identifies people in a group'
{{source-concept: <group>, target-concept: <participant>},
{cardinality: group can have many participants},
{methods: add-participant-to-group, delete-participant-from-group}
{attribute: date-assigned;}
};
relationship: belongs-to, 'identifies groups that people belong-in'
{{source-concept: <participant>, target-concept: <group>},
{cardinality: can belong to none or any number of groups},
{methods: change-group-expertise},
{communication: request-membership:{permission: any-participant}, request-deletion:{permission: any-participant},
{attribute: group-responsibility;}
};
relationship: takes:
{{source-concept: <group>, target-concept: <role-assignment>}
{methods: assign-group-to-role, delete-group-from-role}
{cardinality: group can take many role assignments},
{constraint rule – exclude group member if in groups (group set)}};

In Table 2, we present some example instantiations of the Group Specification fragment for the illustrative collaborative learning system. In this system, each group is a class of one teacher and many students. Since a class can be organized in a number of different ways, we present here different instances of the Group Specification fragment, each representing a different style that a class/group can be specified.

Example instances of Group Specification fragment	**Participant**	**Role Specification** (cf. Table 1)
Single-Manager Group specification (Style 1) Description: This style is suitable when the teacher is also in charge of all groupwork-related responsibilities, e.g. group management, activity management and artifact management responsibilities.	Teacher A	Teacher role Group Manager role Activity Manager role Artifact Manager role
	Students X…Z	Student role
Single-Manager Group specification (Style 2) Description: Instead of assigning separate domain-specific and group-specific roles to Teacher A as in style 1, we can delegate to him/her a single application-specific role "Lecturer-In-Charge", which is made up of all the relevant domain-specific and group-specific roles (cf. Table 1)	Teacher A	Lecturer-In-Charge role
	Students X…Z	Student role
Multi-Manager Group specification (Style 1) Description: This style is suitable when different students take on the responsibilities of activity management and artifact management. Note that different group members can play the same role. For example, Discussion Leader B and Students X to Z play the same "Student" role.	Teacher A	Teacher role Group Manager role
	Discussion Leader B	Student role Activity Manager role
	Noticeboard Secretary C	Student role Artifact Manager role
	Students X…Z	Student role
Multi-Manager Group specification (Style 2) Description: This style is suitable if a single student is in charge of more than type of management responsibilities. For example, a Student Leader X can handle the responsibilities of both an activity manager and an artifact manager.	Teacher A	Teacher role Group Manager role
	Student Leader X	Student role Activity Manager role Artifact Manager role
	Students Y…Z	Student role

Table 2 – Example instances of Group Specification fragment for a collaborative learning system

The group in this case could be a project team with the teacher being the project leader whereas classes and groups become teams responsible for different parts of the project..

5 Summary

This paper has described a set of patterns for describing cooperative work and how they can be used to construct generalized fragments for collaborative work. It has described the semantics of collaboration in terms of a metamodel and used the metamodel concepts to identify groups of concepts that describe common collaborative situations. It has illustrated two such situations in detail, namely, group specification and role assignment, and identified additional potential situations.

Acknowledgements

The work described here was supported by an ARC Discovery grant. We wish to thank the Australian Research Council for financial support.

References

[Br96] Brinkkemper, S.: Method Engineering: Engineering of Information Systems Development Methods and Tools. Inf. Software Technol., 38(4), 1996; pp. 275-280.

[CG99] Carley, K.M.; Gasser, L.: Computational Organizational Theory. In Chapter 7 "Computational Organization Theory. In (Weiss, G. Ed.) Multiagent Systems, MIT Press, 1999.

[Ha96] Hawryszkiewycz, I.T.: Support Services For Business Networking. In (Altman, E.; Terashima, N. Eds.) Proceedings IFIP96, Canberra, Chapman and Hall, London, 1996.

[Ha97] Hawryszkiewycz, I.T.: A Framework for Strategic Planning for Communications Support. In Procs. of The Inaugural Conference of Informatics in Multinational Enterprises, Washington, October, 1997; pp. 141-151.

[Ha05a] Hawryszkiewycz, I.T.: A Metamodel for Modeling Collaborative Systems. Journal of Computer Information Systems, XLV(3), 2005; pp. 63-72.

[Ha05b] Hawryszkiewycz, I.T. An Engagement Modeling Approach for Designing Virtual Organizations. In Proceedings of the International Conference on Information and Communication Technology in Management, Melaka, Malaysia, May 2005; pp. 494-501.

[HBO94] Harmsen, A.F.; Brinkkemper, S.; Oei, H.: Situational Method Engineering for Information Systems Projects. In (Olle, T.W.; Verrijn-Stuart, A.A. Eds.), Methods and Associated Tools for the Information Systems Life Cycle. Procs. IFIP WG8.1 Working Conference CRIS/94, North Holland, Amsterdam, 1994; pp. 169-194.

[He04] Henderson-Sellers, B.; Serour, M.; McBride, T.; Gonzalez-Perez, C.; Dagher, L.: Process Construction and Customization. Journal of Universal Computer Science, 10(4), 2004; pp. 326-358.

[KW92] Kumar, K.; Welke, R.J.: Methodology Engineering: a Proposal for Situation-Specific Methodology Construction. In (Cotterman, W.W.; Senn, J.A. Eds.) Challenges and Strategies for Research in Systems Development. John Wiley & Sons: Chichester,

UK, 1992; pp. 257-269

[LHH05] Lin, A.; Hawryszkiewycz, I.; Henderson-Sellers, B.: An Agent-based Collaborative Emergent Process Management System. In (Bresciani, P.; Giorgini, P.; Henderson-Sellers, B.; Low, G.; Winikoff, M. Eds.), Agent-Oriented Information Systems II, LNAI 3508, Springer-Verlag, Berlin, 2005; pp. 1-18.

[Ra04] Ralyté, J.: Towards Situational Methods for Information Systems Development: Engineering Reusable Method Chunks. In Procs. ISD 2004, 2004; pp. 271-282.

[RP96] Rolland, C.; Prakash, N.: A Proposal for Context-specific Method Engineering. In (Brinkkemper, S.; Lyytinen, K.; Welke, R.J. Eds.) Method Engineering. Principles of Method Construction and Too Support. Procs. IFIP TC8, WG8.1/8.2 Working Conference on Method Engineering, 26-28 August 1996, Atlanta, USA, Chapman & Hall, London, 1996; pp. 191-208.

[RT00] Rehfelldt, M.; Turowski, K.: Business Models for Coordinating Next Generation Enterprises. In Proceedings of the IEEE Academia/Industry Working Conference on Research Challenges, April 27-29 2000; pp. 163-168.

[Sa93] Saeki, M.; Iguchi, K.; Wen-yin, K.; Shinohara, M.: A Meta-model for Representing Software Specification & Design Methods. In Procs. IFIP WG8.1 Conf on Information Systems Development Process, Come, 1993; pp. 149-166.

[SLW04] Shanks, G.; Lansley, E.; Weber, R.: Representing Composites in Conceptual Modeling. Communications of the ACM, 47(7), 2004; pp.77-80.