Prerequisites for Collaborative Problem Solving*

Wolfram Conen and Gustaf Neumann
Information Systems and Software Techniques
University of Essen
D-45143 Essen, Germany

Abstract

In order to answer the question whether the "World Wide Web can serve as an infrastructure for business applications in a globally, distributed and collaborative business environment", one has to step back and address firstly the questions concerning the prerequisites of collaborative problem solving: what is involved in collaborative problem solving in a business context in general and which techniques and what mechanisms available in the Internet can be used to support this task. This paper tries to develop a framework for classification and evaluation of current approaches and to offer perspectives for the development of new infra-structural concepts.

1 Application requirements

Suppose that there exists a socio-technical system $\mathcal{S}_t$ with a set of actors $\mathcal{A}$. The set of actors depends on the actual state[1] of the system $\mathcal{S}$ at a given time t. $\mathcal{S}$ is used if not a specific state is described. A change in the system $\mathcal{S}$ may lead to an updated set of actors $\mathcal{A}'$. An actor may be a human, a robot, a program etc., this will be more precisely specified where needed. A group of actors may be described as single actor, hiding the mechanisms necessary to assign subtasks to individual actors if a tasks is assigned to the abstract actor, representing for example a role or a team.

Initially, there is a problem $\mathcal{P}$ to solve. At a given time t, a problem statement $\mathcal{D}_t$ describes the state of $\mathcal{P}$ in a given language $\mathcal{L}$. The initial mapping of $\mathcal{P}$ to $\mathcal{D}_{t_0}$ is called denotation of a problem. In a business context the problem statement $\mathcal{D}_t$ includes the following information at any time:

- Problem generator a_g, i.e. the actor which first introduces the problem $\mathcal{P}$ into the system $\mathcal{S}$ at the time t_0.
- Problem supervisor a_s, i.e. the actor (or group of actors) responsible for the control of the problem solution process [5].
- Description of the problem to solve.
- Additional problem context information: This information is provided with the intent to guide, accelerate and to ease the problem solution process. The context information can include general information (time and space) and environment specific information such as business rules, problem trail, solution related incentives, reference to a case base etc.

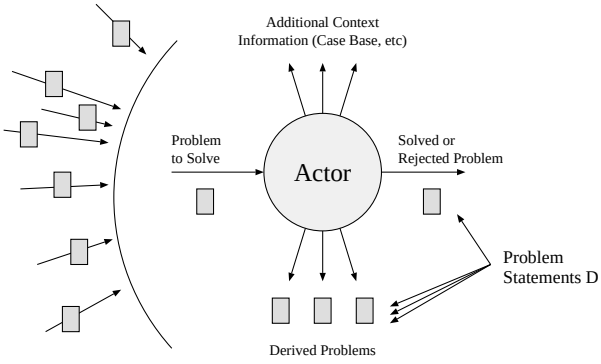

Figure 1: Information flowing to and from an Actor

Now, let us follow the life-cycle $L^{\mathcal{P}}$ of a problem $\mathcal{P}$ through the system $\mathcal{S}$,

$$\mathcal{D}_{t_0} \longrightarrow \mathcal{D}_{t_1} \longrightarrow \ldots \longrightarrow \mathcal{D}_{t_n}, n \in \mathcal{N} \qquad (1)$$

The life-cycle $L^{\mathcal{P}}$ denotes the sequence of changes in a problem statement $\mathcal{D}$. The changes include incorporation of intermediate or final results, refinements of the initial problem statement, progress reports etc. and assignments. For a problem $\mathcal{P}$ to be solved it has to be assigned (see Figure 1) to at least one actor[2] during $L^{\mathcal{P}}$. When a problem

*Published in: Proceedings of WETICE 96, IEEE 5th Intl. Workshops on Enabling Technologies, Stanford, CA, June, 1996

[1] This follows the basic notion of system, as given in [4].

[2] Some problems may be solved without an actor by becoming obsolete; the time "solves" the problem.

is assigned to an actor, the actor may solve $\mathcal{P}$ on its own or he might derive new subproblems $\overline{\mathcal{P}}$ frequently aimed for delegation. The trail $L^{\mathcal{P}}$ of a problem $\mathcal{P}$ consists of the trails of $\overline{\mathcal{P}}$ and $L^{\mathcal{P}}$.

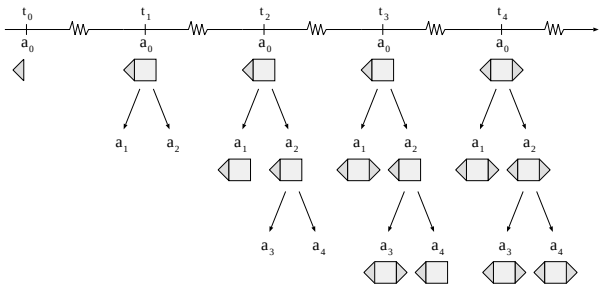

Figure 2: Trail of a Problem

There are two main topics to discuss: (1) the assignment of problems to actors, and (2) the set of operations an actor can perform on (with) a problem. We will concentrate on topic (1) now and switch to topic (2) in Section 1.5 where policies are discussed.

1.1 Actors solving Problems

First, let us suppose that the problem $\mathcal{P}$ is already assigned to an actor a. The assignment took place at time t_a and no update of the problem statement was performed since t_a. a has the problem statement $\mathcal{D}_{t_a}$ available.

The first question a has to answer is the following: *Is the information available in the problem statement sufficient to decide upon the next action to perform?* If the answer is "NO", possible *active* sources of further information are (1) the problem generator, (2) the problem supervisor, or any other actor inside (3) or outside (4) the system $\mathcal{S}_t$.[3] Here, *active* means that the recipient of an information inquiry (which is itself a problem statement) is able to process the statement as described here. Other possible sources of information are all sorts of documents from inside or outside $\mathcal{S}_t$ available to a. Each new information received may contribute to an update of the problem statement $\mathcal{D}_{t_a}$. This task is the first among the main tasks an actor has to perform (Figure 3).

Clearly, at each stage of the possibly iterative task described above, a rough-cut feasibility decision has to made whether the cost involved with the task are still in relation to the incentives which can be deduced from the problem statement $\mathcal{D}_t$. Let us assume that a is autonomous enough to make this feasibility decision itself. Observe that an updated problem statement may lead to less incentives for a

[3] The actor a is itself a possible source of information but a is implicitly included since his knowledge influences/determines the decision upon which action to take next anyway.

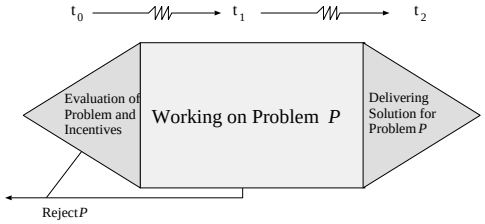

Figure 3: Main Tasks of an Actor

to solve the problem compared to the incentives related to the initial statement $\mathcal{D}_{t_a}$. This cost/value trade-off decision may lead to a break of the information-gathering loop described above. The assignment of a to $\mathcal{P}$ will be resolved, the assignment task has to be performed again. It is a question of the goal system and the control policy whether the costs incurred with the task of information gathering so far will reduce the budget of a or the available incentives of $\mathcal{P}$. We will not discuss this or similar questions further because we only aim at identifying generic task necessary to analyze a problem solution process.

Let us assume that the problem statement is at time t_b specific enough to allow further processing. Most generally, the following operations are now possible:

- Reject the problem. This may have two reason: solving the problem seems to be infeasible or impossible.
- Solve the problem. This involves:
  1. Decomposition of $\mathcal{P}$ into subproblems $\overline{\mathcal{P}}$.
  2. Denotation of the subproblems $\overline{\mathcal{P}}$ and assignment to actors a_x. When the assigning actor is different from a_x a subproblem is delegated. Note that there is a certain inefficiency involved in delegation since it causes overhead such as problem denotation and result integration in a and the problem evaluation and reporting time in a_x. This overhead should be less than the time to solve the subproblem $\overline{\mathcal{P}}$ otherwise the system is locally inefficient.
  3. Integration of results. A solution to a problem can consist of information and/or methods depending on the type of the problem. An example of a method returning problem is "I am looking for a method to solve the following type of accounting problems".
  4. Controlling the problem solution process. This task is important to relate the static problem description to the dynamic environment of the problem solution process. An actor a may obtain information which makes it necessary to re-

vise the problem statement or the decomposition.

- Report the successful or unsuccessful problem solution to the problem supervisor.

The derived problems $\overline{\mathcal{P}}$ can be mutually dependent and might require coordination. These dependencies can be expressed as a sequential or concurrent assignment of $\overline{\mathcal{P}}$ to an actor. The assignment defines the flow of work of the problem.

A decomposition may or may not depend on knowledge about the capabilities of actors. In general a revision of the decomposition is necessary if there is for a given $\overline{\mathcal{P}}$ no assignable actor or if the assigned actor is unable to solve $\overline{\mathcal{P}}$. Assignment revision can make it necessary to revoke subproblems from actors, to change problem statements of $\overline{\mathcal{P}}$ and to communicate these changes, etc. The process of revocation can transitively follow the problem trail. The control subtasks must take all these dependencies into account.

The knowledge and creativity of the actor a is documented mainly in its information processing (while clarifying the initial problem statement, and while controlling the solution process) and in its decomposition and integration decisions. Possibilities to support these decision are described in Section 1.4. The central flow of information related to a problem solution is bi-directional along the problem trail, where problem statements are exchanged. We call it the vertical information flow. The horizontal information flow occurs primarily between competing actors and is discussed in the following section.

1.2 The Assignment Problem

Each time a problem statement $\mathcal{D}$ is initially formulated, an actor $a \in \mathcal{A}$ has to be assigned subsequently to it for further processing[4]. This assignment problem can be tackled in various ways. It is a problem specific formulation of the classical resource allocation problem [7].

1. *Explicit Assignment:* An actor a_1 assigns the problem explicitly and directly to another actor a_2. a_1 must be sufficiently informed about the capabilities of a_2. If a_2 is not allowed to reject or delegate the problem, a_1 has to consider a_2's *workload* as well.

2. *Competitive Assignment:* The problem statement $\mathcal{D}$ is *offered* to one or more actors. This may include the use of a blackboard or similar tool. One or more actors may/should compete for the assignment of the problem.

The second case mentioned above leads to horizontal information flows. Each competing actor may gather information to specify the problem statement more precisely to decide whether he is able to solve the problem within the cost/incentive boundaries given in the problem statement.

In order to decide which actor should work on a problem, coordination among the actors is necessary. *Implicit coordination* is a process where the problem supervisor collects information about the abilities of the competing actors, (e.g. time, cost, and quality information) and performs the assignment based on this information[5]. The information flows between the supervisor and each competing actor.

Explicit coordination refers to a process of solving the assignment problem through direct communication among competing actors. There is no inherent need for information flow between the supervisor and every single actor. The competing group of actors solves the assignment problem on its own, considering individual workloads, organizational relations, or strategic decisions. This explicit coordination enables the problem supervisor to put the problem statement on a blackboard and to "take no care" of the assignment process (while still having the duty to control the problem solving process).

1.3 Information Flows

Let us summarize the information flows concerning $\mathcal{P}$ in $\mathcal{S}$. These information flows are of the following types:

1. Information flows between actors following the problem trail vertically (up and down). Problem statements are exchanged, the problem solution process is controlled. In the case of competitive assignment bidding and knocking down information is exchanged as well.

2. Information flows between actors on the same level of the trail. This horizontal information exchange happens for example during competitive assignment, in team formations, or for synchronizing mutually dependent subproblems.

3. Information flows between one actor and some internal or external information sources. This *supplemental information flow* may occur at each stage of

[4] If an actor d has to decide at a certain point in time, t, which actor a should be assigned to a problem, obviously the set of actors $\mathcal{A}$ available for assignment depends on $\mathcal{S}_t$. But this is only part of the truth: if the rules encoded in $\mathcal{S}_t$ allow for assignments of problems to actors from "outside" $\mathcal{S}_t$, the set of actors need not be bound to elements of $\mathcal{S}_t$. Nevertheless, this may lead to an extension/change of $\mathcal{S}$ such that a become an element of $\mathcal{S}$ at a subsequent point in time.

[5] This process can be a market-like process of sending bids and price information; for some background see for example [7].

the problem solving process, e.g., while trying to create a more precise problem statement, while deciding whether to solve the problem or not, or while trying to solve the problem. Candidates for supplemental sources are case bases, data bases, regulations etc.

1.4 Learning about Problems

When a problem $\mathcal{P}$ is assigned to an actor, the actor should be supported by providing supplemental information about previously solved problems $\mathcal{P}_s$ where $\mathcal{P}_s$ should be sufficiently similar to $\mathcal{P}$. In this task it is necessary

1. to identify and collect relevant information about related problems (a case base). It may suffice to simply store the problem trail, but only if the decisions leading to decompositions of problems are simple and need no documentation. Otherwise, an actor has to explain his decomposition and integration decisions to capture his knowledge;
2. to recognize similarities between problems (cases). This generally includes the ability to group/abstract problems into types;
3. to offer an interface to actors for easy search and evaluation of cases.

Later in this paper (Section 2) the contribution of problem trails to the case base is called "indirect collaboration".

1.5 Policies in Systems for Problem Solving

The constituents of a system for supporting problem solving in a business context can be described in the terms given above, with one addition: policies.

A *policy* determines the choices to be made while designing and using an arbitrary system, for example, an assignment policy for the assignment of problems to actors. In general, a policy is a function of some specific information (like a problem) and a context to operations, e.g.

$$p(\textit{Problem statement}, \textit{Context}) \longrightarrow \textit{Operations} \qquad (2)$$

If the operations are unknown at the time of the definition of the policy, the function is not to operations but to criteria allowing a testing of operations for suitability. Among the policies needed to constitute a system for problem solving, are the following.

Assignment Policies These are policies related to the way problems should be assigned to actors.

- Explicit assignment of $\mathcal{P}$ through a_s to a. Assignment only along the management hierarchy.
- Competitive assignment. Either a_s selects a competing actor (a) or the group of actors decides itself about the assignment (b).
- Policies stating to whom certain problems in certain circumstances may be assigned to, for example, such a policy determines who will become the supervisor of newly generated problems.

Control Policies This may refer to

- Control of the problem solution process, such as stating whether direct addressing of actors two or more management levels above a_s is allowed, or what measures can be taken if a responsible actor a_1 obtains solutions from a_2 in bad quality (insufficient, to late ...).
- Control along the management structures of $\mathcal{S}$ (and thus somewhat independent of problem trails), stating, for example, whether it is allowed for a manager a_1 of an actor a_2 to force a_2 to reject a competing problem.
- Control of the structure of (parts of) the system $\mathcal{S}$, stating who is eligible to change certain policies, goals, elements of $\mathcal{S}$.

Information Policies This may refer to

- Policies regulating which actor is allowed in which situation to communicate to certain other actors (from in- or outside $\mathcal{S}$) in which manner, e.g. members of the department x are not allowed to request information from a member of some department y while solving problems of a certain type.
- Protocol policies regulating the type of information flowing in certain communication contexts.

Problem Solution Policies This may include regulations about the amount of work an actor a has to invest into a subproblem before a is allowed to delegate the subproblem as well as policies regulating the access to supplemental information such as the problem case base. The policy defines what is retrievable, what has to be contributed etc.

Applying Policies The following policies may be part of a description of a hierarchical organization structure.

- Assignment policies: Explicit assignment. Only certain actors are allowed to assign problems between departments or to actors outside $\mathcal{S}$ (out-sourcing). Generally, assignment is allowed downwards along the management hierarchy.

- Control policies: Problems may exist without an explicitly assigned supervisor. A problem supervisor a_1 may directly control the problem solution of any actor a_2 below a_1 in the management hierarchy. Arbitrary measures may be taken if a_1 expects a solution of bad quality. No direct control across department boundaries is possible. In this case, control is only possible via a manager of a_2. In general, the right to control structural elements of $\mathcal{S}$ is given to certain actors with a certain position in the management hierarchy. Thus organizational development, which necessarily involves the change of structural elements of $\mathcal{S}$, can only be performed through a complex process always involving certain actors.

- Information policies: Communication is often restricted by department boundaries. Even for problem supervisors is it impossible or difficult to communicate with all actors on the trail of their problem.

- Problem solution policies: No generally accessible case base exists. Actors not allowed to further delegate problems have to solve the assigned problem on their own. Delegation (and thus decomposition) is problem-independently restricted by the assignment policies.

A policy-based characterization of an organization can be helpful in analyzing information flows, information types, communication and supervising structures etc.[6] Now, the set of elements necessary to describe systems trying to solve problems is sufficiently complete. The following sections discuss some general requirements for information systems supporting problem solving and compare these requirements to the capabilities of Web-based systems.

2 Collaborative Problem Solving

In this context, collaboration refers to a *direct*[7] or *indirect* process of intertwined activities of two or more actors while solving the same or related problems[8]. These activities may or may not require explicit coordination.

The simplest form of collaboration is indirect collaboration, where actors use results/knowledge/information other actors made available for example in a shared case base.

[6] The reader may try to develop a set of policies describing a collaborative, team-oriented organization structure.

[7] Due to the valid criticism in [2] we do not use the categories "asynchronous" and "synchronous" to categorize collaboration.

[8] If a general business goal is the "point of relation" between those problems, the use of the term collaboration becomes meaningless because if discussed from this perspective all problems in a business context require collaboration to be solved.

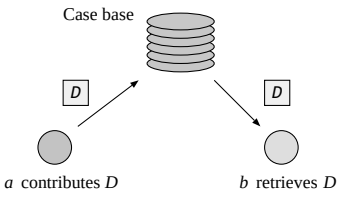

Figure 4: Indirect Collaboration

Direct collaboration requires a commitment of all participating actors. As shown in Figure 5, a delegated collaboration leads to contributions to the problem solution of the delegator.

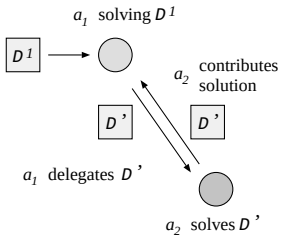

Figure 5: Direct Collaboration by Delegation

In the case of a team collaboration the exchange of problem statements is simplified. In general, the transaction costs [8] related to the collaboration of the team members decreases, thus enabling an intensified flow of problem statements. A second effect diminishes the required problem statement size. Information concerning the problem context can be reduced since this information is generally exchanged in the phase of team formation.

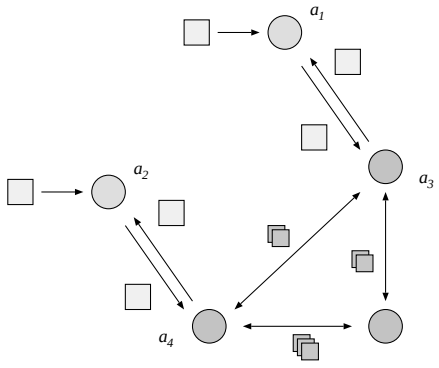

Figure 6: Direct Collaboration by Team Formation

The notion of collaboration does not add basic elements for problem solving systems to those already discussed. Instead, collaboration can be described as sending problem

statements back and forth between collaborating actors or actors willing to collaborate.

3 General requirements and the Web

Below, some examples of relating the elements of a general system for problem solving to general requirements and to a realization of those requirements on top of a Web-based infrastructure are given. Here, *infrastructure* stands for the basic elements to construct and operate the system. The requirements mentioned below are focusing on the problem statements $\mathcal{D}$, which are the central elements of systems for problem solving.

Certainly, other requirements to the infrastructure can be derived by focusing on the other constituents of $\mathcal{S}$ (actors, assignments, policies, control and coordination).

3.1 Problem Templates

Requirements: State info, descriptive part, active part supporting abstraction and definition of the problem.

Relation to the Web: HTML documents + state info + general and (added while solving the problem) problem class specific methods.

3.2 Distributing Statements

Requirements: Sending statements to actors or groups of actors (addressing) or to blackboards.

Relation to the Web: Mailing of problem statements (complexity depends on the way, problem statements are maintained. If they are implemented as active objects, including methods to update the statement or to compute the solution, an active mail system seems to be a suitable solution, see [1, 6]). Posting to Newsgroups.

3.3 Update Problem Statements

Requirements: Depending on the problem class or more general, the state of the problem statement, different update methods may apply, thus requiring a context and problem specific treatment.

Relation to the Web: Flexible problem statement dependent reactions of a Web Browser needs the transition of more semantic information. Easiest way to do so is the use of a scripting language. This keeps the document open to behavioral changes (if a received problem statement $\mathcal{D}_{t_i}$ is viewed as a template for the generation of the next problem statement state $\mathcal{D}_{t_{i+1}}$).

3.4 Communicating the Update

Requirements: Sending of statements to problem supervisor and/or an actor responsible for maintaining the problem statement or to all actors currently interested in the state of the problem statement.

Relation to the Web: Keeping the above mentioned update in mind, this means that either $\mathcal{D}_t$ is mailed in form of an active document to the supervisor (performing the update mostly automatically) or that the document $\mathcal{D}_t$ is placed on a server and a pointing information (URL) is sent to the supervisor.

4 Analyzing Infrastructures

These listing of the requirements above are far from complete, but they show the direction which a constructive and sufficiently complete analysis of an application area may follow: (1) Determine the general characteristics of the area, (2) analyze these characteristics and relevant examples to deduce a set of requirements, (3) check which requirements can be fulfilled with the infrastructure the chosen implementation environment offers, and, finally, (4) specify which enhancements have to be made to this implementation environment to meet the requirements of the application area. Further results of an analysis of the capabilities of Web-based problem solving applications will be presented at the workshop.

References

[1] N.S. Borenstein: “Email With A Mind of Its Own: The Safe-Tcl Language for Enabled Mail”, ULPAA 1994 Conference Proceedings (1994).

[2] A. Dix, J. Finlay, G. Abowd, R. Beale: Human-Computer Interaction, Prentice Hall (1993).

[3] R.D. Eckert, R.H. Leftwich: The Price System and Resource Allocation, 10th Edition, The Dryden Press (1988).

[4] P. A. Fishwick: An Integrated Approach to System Modeling. ACM Transactions on Modeling and Computer Simulation, Volume 2, No. 4, (1992).

[5] W. Hussy: Denken und Problemlösen, Verlag Kohlhammer, Stuttgart (1993).

[6] R. Mühlbacher, G. Neumann: Towards a Framework for Collaborative Software Development of Business Applications, in review for publication in the WET-ICE’96 proceedings.

[7] M.P. Wellmann: “Market-Oriented Programming: Some early Lessons”, in: S.H. Clearwater: Market-Based Control, World Scientific (1996).

[8] O. Williamson: “Transaction-cost Economics: the Governance of Contractual Relation”, in *Industrial Organisation, The International Library of Critical Writings in Economics*, Ed. Oliver Williamson (1990).