Learning to Reduce Interorganizational Learning: An Analysis of Architectural Product Innovation in Strategic Alliances

Roman Grunwald and Alfred Kieser

There is wide agreement in analyses of strategic alliances that, regardless of the purpose of the alliance, members of the partner organizations should engage in intensive mutual learning to make the alliance a success. In contrast to this view, the present article shows that in strategic alliances aimed at product innovations by recombining partners’ extant technologies, learning between specialists can be reduced considerably without jeopardizing success. This is made possible through four interconnected mechanisms integrated into the concept of transactive organizational learning (TOL): (1) modularization, which allows specialists of different domains to develop modules to a large extent independently of each other and to concentrate communication between themselves on the design of interfaces between modules; (2) storing of knowledge in artifacts instead of in organizational members’ memories; (3) localization of knowledge not present in the project team but for which a need has arisen through transactive memory; and (4) knowledge integration by prototyping (i.e., by repeated testing of modules and of interactions between modules until a satisfactorily working end product is achieved). Although these four mechanisms reduce the need for cross-learning between specialists of different domains, some common knowledge and some cross-learning between the partners’ specialists is still required. Case studies on four of SAP’s strategic alliances for product innovation with different partners lend empirical support to this study’s concept. The article concludes with implications for practice: Companies should find out whether the TOL mechanisms that reduce time to market are present, to what extent their potential is exploited, and how well they work together.

Is Interorganization Learning a Requirement for Success?

Companies cooperate with other companies because their managers are convinced that through cooperation they are able to attain ends they could not achieve alone, or at least not as quickly (Borys and Jemison, 1989; Dunning and Boyd, 1997; Hergert and Morris, 1988; Ireland, Hitt, and Vaidyanath, 2002; Nohria and Garcia-Pont, 1991). If such a cooperation is projected for a longer time span, it usually takes the form of a strategic alliance, which can be defined as an “arrangement between two or more independent companies that choose to carry out a project or operate in a specific business area by coordinating the necessary skills and resources jointly rather than operating alone or merging their operations” (Dussauge and Garrette, 2000, p. 99). Product innovation can be assumed to be a frequent purpose of strategic alliances since, as Simonin (1999, p. 595) argues, strategic alliances “constitute perhaps the most adequate but nevertheless

We would like to thank Christian Czernich, Manfred Fuchs, Ulrich Koch, Till Mettig, Achim Oberg, Markus Reihlen, Christina Schmickl, Martin Schulz, Sven Weißmann, and two anonymous reviewers for their helpful comments. Support from the Deutsche Forschungsgemeinschaft is gratefully acknowledged.
challenging vehicle for internalizing the other’s competency” for pooling knowledge in a race against rivals and time. Recently, the pooling of knowledge about technologies between alliance partners has increasingly attracted the interest of management researchers (Hagedoorn, 1993; Tidd, 1995). (Although this kind of cooperation does not require strategic, or long-term, alliances, the present study uses this term in accordance with most authors).

Assuming that a common knowledge base increases the likelihood of success, many authors argue that the partners should engage in intensive cross-learning. Usually, this assumption is supposed to apply to all kinds of strategic alliances. For example, Lubatkin, Florin, and Lane (2001, p. 1354) contend that, as a general condition for the success of strategic alliances, “the partners must not only learn themselves and improve their respective knowledge base, but also learn to learn together, and learn how to exploit this new knowledge that makes them interdependent.” Referring to Nonaka and Takeuchi (1995), they argue that in such an interorganizational cooperation “individual and collective knowledge-structures co-evolve, each influencing and being influenced by the other, through social interaction and embedded in the firm’s idiosyncratic social fabric” (p. 1356). They draw on psychological concepts of learning between individuals—more precisely, of learning between students and their teacher—to develop an evolutionary model of reciprocal learning alliances in which organizations gradually have to learn to “co-experiment and jointly discover” (p. 1365). Though Lubatkin, Florin, and Lane (2001, p. 1367) concede that the objective of reciprocal learning alliances is usually not to absorb as much of the respective partner’s specialized knowledge as possible, they list five basic factors as requirements for successful reciprocal learning, among them the following ones: (1) the partners must already possess “a basic awareness of the semantics, episodes, and the articulate cause and effect linkages for decision rules, that ground each other’s knowledge structures” (p. 1366); (2) the partners should be able to speak the basics of each other’s language as well as have some proprietary knowledge of their respective technologies; and (3) the partners must “think and behave in similar ways” (p. 1368).

The importance of mutual learning is also contained in the concept of absorptive capacity (Cohen and Levinthal, 1990), to which many authors refer when analyzing performance of strategic alliances (Barringer and Harrison, 2000; Inkpen, 2000; Kumar and Nii, 1998; Lane and Lubatkin, 1998; Mowery, Oxley, and Silverman, 1996; Parise and Henderson, 2001; Reid, Bussiere, and Greenaway, 2001; Van Den Bosch, Volberda, and De Boer, 1999). Absorptive capacity expresses an organization’s ability to assess, appropriate, and exploit knowledge and is seen as depending to a large extent on the company’s motivation to learn from its partners, its partners’ cultures—including the incentive structure—and its partners’ technological competencies, especially the qualification of their employees.

Child (2001, p. 656) also emphasizes intensive interpartner learning as a precondition for the success of strategic alliances: “This process [of interorganizational creation of new knowledge] implies that mutual learning occurs through a constructive integration of the different inputs offered by the partner and their members.” Learning between the partners “involves the retrieval of knowledge that has been created in the collaborative units and its internalization in the parent firms” (p. 659). Referring to Nonaka and Takeuchi (1995), Child calls for open communication and an intensive exchange of relevant knowledge, even if knowledge sharing is not necessitated by current projects: “For learning to take place, information or a concept available to one person or group needs to be shared by others who may not need it immediately” (p. 676).

In contrast to the authors who plea for an unconditionally intensive interpartner learning in strategic alliances, the present study argues that the required intensity of interpartner learning is contingent on the goals of the alliance. If the goal is exploitation of
partners’ extant technologies for product innovations, only a modest level of interpartner learning is required. Hamel (1991, p. 84) formulated an early version of this proposition when drawing a “crucial distinction between acquiring such skills [that underlie global competitiveness] in the sense of gaining access to them and actually internalizing a partner’s skills.” Inkpen (1998, p. 72) also observes, “In some alliances, partners aggressively seek to acquire alliance knowledge while in others, the partners take a more passive approach to knowledge acquisition.” In a similar vein, Koza and Lewin (1998), building on March’s (1991) concept of exploitation and exploration, differentiate between strategic alliances that aim at achieving specific performance objectives or revenue enhancement by exploiting existing technologies on the one hand and strategic alliances that aim at exploring new technologies on the other hand. They argue that alliances for the exploitation of extant technologies do not necessitate interorganizational learning of a more intensive nature.

Also building on the concept of exploitation and exploration, Grant and Baden-Fuller (2004) argue that knowledge acquisition is of minor importance for certain cooperations aiming at knowledge exploitation. Daimler-Benz’s partnership with Swatch in designing the “Smart” is an example for such a cooperation since it was not motivated by Mercedes’s desire to acquire Swatch’s knowledge in precision engineering nor by Swatch’s desire to absorb Mercedes’s knowledge in car manufacturing. Rather, it was “both partners’ desire to create value through combining their separate knowledge bases” (Grant and Baden-Fuller, 2004, p. 65). The same kind of knowledge sharing took place when Pavarotti arranged a joint project with the Spice Girls: This cooperation was certainly not motivated by Pavarotti’s wish “to be a girl band or the Spice Girls acquiring operatic skills” (ibid.). Consequently, Grant and Baden-Fuller distinguish between knowledge generation, based on the goal to jointly explore the partner’s knowledge, and knowledge application, which aims at jointly exploiting the partners’ knowledge without sharing it.

A categorization of innovations that is related to the distinction between exploitation not requiring intensive knowledge sharing and exploration requiring it is that of architectural and modular innovations (Henderson and Clark, 1990). An architectural innovation is the result of changing “the way in which the components of a product are linked together, while leaving the core design concepts and thus the basic knowledge underlying the components untouched” (Henderson and Clark, 1990, p. 10). Modular innovations, in contrast, involve the implementation of a new technology that overturns the core design concepts while leaving the established linkages between the components untouched. This article focuses exclusively on architectural product innovations that are pursued in alliances. It is argued that in these innovations partners are usually not interested in sharing knowledge to a greater extent. Their intention rather is to exploit their knowledge bases to economize on resources and to speed up time to market. Innovations of this kind already claim a large share of the product innovation market and are more and more common (Cebon, Hauptman, and Shekhar, 2002; Hagedoorn, 1993; Tidd, 1995).

The Concept of Transactive Organizational Learning

Transactive organizational learning (TOL) is a learning mode that allows specialists cooperating in strategic alliances to exploit the knowledge of other specialists without having to share a considerable quantity of knowledge with these other specialists. The present article argues that an intensive exchange of specific design knowledge between the partners’ specialists involved in architectural innovations would conflict with the fundamental organizational principle of specialization, which is conditional on the bounded rationality of the human brain. Attempting to broaden their knowledge domains specialists will, sooner or later, find out that their personal absorptive capacity is limited—that there is a trade-off between depth and breadth of knowledge. Because of these cognitive limitations, an organization has to be understood “as a machinery for coping with the limits of man’s abilities to comprehend and compute in the face of complexity and uncertainty” (Simon, 1979, p. 501). Therefore, in contrast to the authors who advocate intensive cross-learning, this study’s basic research question is not how specialists from the partner organizations can learn as much as possible from each other. This would mean partial despecialization and would thus be in conflict with bounded rationality; rather, how can strategic alliances organize architectural innovations in such a way that their specialists do not have to exchange large quantities of knowledge? In the words of Demsetz (1991, p. 171), “Although knowledge can be learned more effectively in specialized fashion, its use to achieve high living standards requires that a specialist..."
somehow uses the knowledge of other specialists. This cannot be done only by learning what others know, for that would undermine gains from specialized learning.” The mechanisms that lie behind the somehow in Demsetz’s quote is what is of concern in this article.

TOL consists of four interacting mechanisms.

Modularization

Modularization means that larger entities (i.e., products, processes, organizations)—in this case projected architectural innovations based on alliance partners’ technologies—can be decomposed into less complex components (i.e., modules) (Baldwin and Clark, 1997). These can be designed or, in cases of changes of existing systems, redesigned by specialists or groups of specialists who work relatively independently from each other on their respective modules. This approach significantly reduces coordination and production costs as well as time to market: “If all components must be tightly integrated and optimized for each other, their production often requires that all individuals involved in such design and production also work in close contact. A modular product design, in contrast, can enable the production process to be decentralized” (Schilling, 2000, p. 320). Decentralization here means delegated to loosely coupled teams of specialists. To the extent that a correspondence exists between the modularity of the projected architectural innovations and the organizational structure of the partners’ research and development (R&D) area, alliance partners’ teams can redesign modules of their technologies in their “home organizations” (Galunic and Eisenhardt, 2001; Sanchez, 2000; Tidd, 1995).

Another advantage of modularization is that it can increase scope and speed of innovation (for an exhaustive overview of the advantages of modularization see Sanchez, 1999). The argument is that it is easier to rearrange or to recombine modules into new technologies than to design new technologies in a nonmodularized style (Dosi, 1982; Hsuan Mikkola, 2003; Sanchez and Mahoney, 1996). Recombinations of modules result in architectural innovations—the kind of innovations dealt with in this article.

Knowledge Storing in Artifacts as an Effective Storing Mechanism

Knowledge goes into artifacts and is captured in them or, as Hedlund (1994, p. 79) says, “[A] tangible product is knowledge in a highly articulated form.” An artifact (i.e., a module) represents, or contains, knowledge that its designers applied in designing it. Persons who do not possess this knowledge can still use the artifact and can learn many things from it, even though they are not able to construct it themselves. For example, one can use a computer to develop a program and to get it running without knowing how to design a computer. Specialists involved in architectural innovations between strategic partners will, in a first step, assess the capabilities of the modules of the two technologies with regard to the new product. For example, if the producer of a mobile phone and a producer of an MP3 player decide to develop a mobile phone with an integrated MP3 player, they will probably start their cooperation by exchanging specifications of the components that they intend to integrate into the new product. For many components this information is readily available as they already are components of different products of the partners’ portfolios.

Groups of specialists who work on specific modules need not possess knowledge on how to design modules for which they are not responsible. In principle, it is sufficient for them to receive information on the functionalities of the respective other modules with which their modules are supposed to interact. They get this information through written descriptions, presentations of the other module groups, or experimentation with these other modules. When groups of specialists present possible solutions for modules they are working on to groups working on other modules, they usually concentrate on information needed by the respective other groups to adapt their module to the input–output requirements of modules with which their module is supposed to interact.

A Mechanism for Locating Knowledge that Is Not Available within the Team

When an architectural innovation is started, it is usually not too difficult to identify the specializations that should be represented in the project team. However, it cannot be ruled out that problems will emerge in the course of the project necessitating expertise not available in the project team. In this case a mechanism is required to localize this needed knowledge within the alliance organizations or in other organizations. Transactive memory is such a mechanism (Wegner, 1987, 1995). Applying this concept to product innovation in strategic alliances means that it
is not necessary for members of the project group to know everything relevant for the project right from its start; it is sufficient to possess a directory of this knowledge: “Our directories for memories held by others can be thought of as metamemories. That is, they are memories about memories” (Wegner, 1995, p. 326). In this way, a “knowledge-holding system [is created] that is larger and more complex than either of the individuals’ own memory systems” (Wegner, 1987, p. 189). Systems of this sort become manifested, for example, in a company’s yellow pages, which serve the purpose of identifying organizational members or external experts with specific knowledge (Marchand and Davenport, 2000). As Walter and Ritter (2003) and Ritter and Gemünden (2003) demonstrate, including customers in directory knowledge is likely to contribute to value creation in product innovation processes.

Prototyping

If a group working on a certain module learns about other modules, in most cases a perfect fit cannot be achieved since knowledge stored in artifacts that specialists are not able to design themselves is necessarily imperfect. Descriptions by specialists who are able to design the respective other modules are attempts to communicate what these specialists think other groups of specialists need to know to be able to design their modules so that these interact successfully with other modules. The members of other groups can ask questions in their attempts for clarification. But, to a certain extent, these are discourses between groups of specialists who draw on different theories and use different terminologies. The problem becomes aggravated if designs of interacting modules are not finalized yet, which means that the specialists are presenting moving targets to each other. Design for module A is dependent on the design of module B and possibly design of module B on that of module A. These processes of mutual adjustment have to be organized under the condition that specialized knowledge in one group should not be shared to a larger extent with other groups.

Prototyping provides a solution to tricky problems of this sort (Lichter, Schneider-Hufschmidt, and Züllighoven, 1994; Sanchez, 1996; Schrage, 1993). In essence, prototyping is a process based on trial and error rather than on completely rational design. Thus, with the number of prototyping rounds the design of modules constantly improves in a way that secures coordination between them. In its simplest form it is present when groups of specialists working on different modules present each other their design concepts, whereby they concentrate on input-output requirements between modules and ask for feedback. For example, one group tells the other, “It would make our job much easier if your module could provide an absolutely stable pressure of x bar.” In later stages, compatibility and functionality are tested in more rigorous forms of prototyping, such as by applying computer models that simulate the performance of individual modules or of a number of coupled modules. The last stages of prototyping consist of coupling groups of completed real modules for testing, modification, retesting, and so forth until the entire real product can be tested. The trick with prototyping is that problems encountered in tests lead to modifications of interacting modules that still do not require intensive interspecialist learning. Only educated guesses for modifications are required. These can be generated on the basis of prototyping results: For example, “It will not work this way, you will have to increase pressure and keep it absolutely stable.” The other group replies, “Perhaps you guys can do something to reduce losses of pressure in the hydraulic system of your module.” Modifications in the agreed-on directions are then tested in further prototyping runs, and new modifications are carried out until the results are satisfactory.

The Role of Common Knowledge

The TOL mechanisms reduce the need for cross-learning between specialists. However, this does not mean that there is no need for knowledge sharing. As Grant (1996a, 1996b), from whose concepts this section draws, points out, knowledge sharing between specialists can in principle increase the effectiveness of their coordination activities. Two kinds of common knowledge have to be distinguished: common coordination knowledge and common content or design knowledge. Common coordination knowledge concerns knowledge that supports specialists of different domains in their coordination efforts without being attributable to any specialized product design knowledge. Common content knowledge applies to overlapping specialized product design knowledge between specialists of different domains. The specialists share knowledge about technical details of components.
A common natural language such as English or German is a basic element of coordination knowledge. Members of cooperating partners or departments should be able to communicate with each other. The effectiveness of their coordination could be further increased if they also shared artificial languages such as programming languages. For example, knowledge about musical notation and the conductor’s standardized movements reduce the communication required between the members of an orchestra (i.e., specialists in different instruments) when rehearsing a concerto. Flowcharts or abstract programming languages provide the same function for coordination between members of software engineering teams. Furthermore, powerful programming languages allow an efficient definition of and communication about interfaces (Sommerville, 2001).

With regard to common content knowledge, Grant (1996b)—without applying a distinction between common coordination and content knowledge—maintains that some overlap or commonality of specialized content knowledge between different groups of specialists also facilitates cooperation. He admits, though, “There is something of a paradox in this . . . If two people have identical knowledge there is no gain from integration—yet, if the individuals have entirely separated knowledge bases, then integration cannot occur beyond the most primitive level” (p. 116).

Neither common coordination nor content knowledge is a subconcept of the TOL concept. Common content knowledge is the result of cross-learning; it is not a mechanism for reducing cross-learning like the other subconcepts of TOL. It would also be misleading to characterize the TOL mechanisms as coordination mechanisms. By reducing the need for cross-learning, the TOL mechanisms facilitate coordination; they do not bring it about. Coordination devices such as hierarchy, operating procedures, plans, and ad hoc communication between interacting individuals or groups are still needed for coordination in product innovation alliances as well as in other organizational activities.

**Deriving Propositions from the TOL Concept**

The following sections derive testable propositions from the TOL concept and the concept of common knowledge.

**Reducing the Need for Cross-Specialist and Cross-Partner Learning through Modularization**

An architecture for a modularized product design, according to Ulrich and Eppinger (2000), consists of (1) the descriptions of how elements’ functionalities have to be arranged so that they contribute to total product performance; (2) the assignments of functional elements to physical components (i.e., a description of which subunits bring about which functions); and (3) the specification of the physical components’ interfaces. Most architectures for modularized systems lie on a continuum between the extremes of fully modular and almost fully integral (Brusconi and Precice, 2001). The looser, or tighter, the coupling between subunits and the higher the degree to which the system architecture enables, or inhibits, the mixing and matching of components, the higher, or lower, the degree of modularity is (Schilling, 2000). There does not exist, however, an optimal degree of, or an optimal solution for, modularization since designing architectures for modularized products is a creative process: “Choices of modules are guesses about appropriate decompositions” (Ethiraj and Levinthal, 2004, p. 172). Attempts to technically optimize the architecture for modularization may get into conflict with the goals to minimize effort and time to market.

The authors who point out that modularity can reduce coordination needs in and between organizations do not describe in detail how teams of specialists from different organizations who work on different modules coordinate their work under the condition of limited knowledge sharing. In particular, they do not discuss how much, or how little, cross-learning between specialists is necessary and how a reduction of cross-learning is accomplished. This is one of the basic questions this article attempts to answer. Three different kinds of modules in architectural product innovations are distinguished (Figure 1): (1) modules that remain unchanged or are not needed in the intended new product; (2) modules that have to be modified in accordance with changes in their interfaces with other modules but that do not require changes in their basic functionalities; and (3) new modules that are needed to provide functions not yet present in the technologies that are recombined. For an exemplification, return to the example of a mobile phone with an integrated MP3 player. Some modules of the mobile phone and the MP3 player may not require changes. Some, such as the panel of the MP3 player or the earpiece of the mobile phone, probably will. Some new modules may
have to be added, such as a relay that stops the MP3 player when a message comes through. It can further be assumed that the second type of modules can largely be constructed by individual specialists or by teams of specialists who were responsible for the construction of the original modules in one of the partner organizations—that is, by specialists who need not acquire fundamentally new knowledge for this task. The third type of modules—and only those—is likely to require some learning across specialists of different domains and across the partner organizations. Therefore, if having more than the minimum number of the third type of modules does not lead to a significant advantage—either for the producers or the customers—it can be assumed that the designers of the new product’s architecture, attempting to save on resources and time, will try to minimize the number of these modules. This can be achieved by leaving as many modules as possible intact or subjected to only minor changes and by adding new modules that add on new functions to the old modules with whom they are linked. With regard to the criterion “effectiveness” or “elegance” a different design might have been chosen. It is like adding annexes to an existing house. The goals of the inhabitants are roughly met. Given more time and money, however, they would have preferred a more radical remodeling or even a new building with a more effective layout. The goal of architectural innovations is not to maximize innovativeness but to exploit the extant technologies as effectively and as quickly as possible by finding a recombination of modules that fulfills the functionalities specified for the new product. Increasingly, software developers try to built flexibility for integration into the architecture—that is, design an architecture that consists of standardized modules with standardized interfaces (Karch and Heilig, 2005). Coming back to the house metaphor, houses that consist of standardized elements can easily be assembled and reassembled into houses that fit different user needs.

These considerations lead to Proposition 1.

**P1:** In product innovations that come into existence through the exploitation of alliance partners’ extant technologies, the necessary level of interorganizational and interspecialist learning can be reduced through modularization. Modularization enables the majority of groups of specialists to work separately on modules and to cooperate with each other while sharing only modest amounts of their respective specialized knowledge bases.

**Identifying Relevant Knowledge through Transactive Memory Systems**

If, during the course of a project, problems emerge that cannot be solved in the team, transactive memory—the
directory knowledge of organizational members—is likely to be activated to identify experts within or outside the alliance who possess the required knowledge.

P2: In architectural innovations in strategic alliances, transactive memory is used as a mechanism to localize required knowledge not represented in the project team.

Storing Knowledge in Artifacts and Recombining It through Prototyping

As argued previously, prototyping is the basic mechanism for recombining technologies for architectural product innovations. Different forms of prototyping experiments like small-scale tests, computer simulations, or rapid prototyping can be distinguished (Thomke, 1998). The present study maintains that critical evaluations of a new concept, presented by a group working on module A to another group working on module B with which module A interacts, constitutes a specific form of prototyping. By listening to their mutual presentations the groups test whether modules that have to interact with modules they are working on provide the functions they are supposed to provide. The information exchanged in these presentations predominantly concerns properties of interfaces and is, therefore, less voluminous than the knowledge needed to design the different modules.

When a group of specialists discusses the feasibility of other groups’ proposals, their comments and suggestions reflect their experiential knowledge (March, 1991). Thus, feedback on a new concept or prototype that initiates modifications can be provided through—if necessary, repeated—assessments by organizational members not involved in the design of a specific module. They can investigate whether the presented concept or prototype of modules, subsystems of modules, or the total system will function in the intended way. However, each group keeps the responsibility for the design of its module. This means that an extensive sharing of design knowledge is not required. In later stages of the project, more sophisticated tests, such as computer simulations, are applied. These thoughts lead to the following propositions.

P3: In architectural innovations in strategic alliances, knowledge going into a module is stored in concepts and prototypes—artifacts—rather than in the project team members’ memories. Thus, project team members can take this knowledge into consideration when working on their own modules without having to be able to reproduce it in detail.

P4: In architectural innovations in strategic alliances, groups working on modules recombine knowledge, to a great extent, by critically assessing each other’s concepts and prototypes and by performing more powerful trial-and-error experiments (i.e., prototyping) of the modules’ functionalities. Critical feedback on concepts and prototypes triggers revisions and new tests. Thus, after a limited number of such exercises, the modules can be assembled into an end product that is subjected to prototyping until the product functions satisfactorily.

Common Knowledge Supports Coordination

Kale, Dyer, and Singh (2002) report that in companies that have accumulated a lot of experience in strategic alliances, usually a number of approaches and instruments develop that result in a dedicated alliance function. This common knowledge consists of guidelines, manuals, tools, and templates. More broadly, it includes processes for the identification of potential partners and the evaluation of their resources and capabilities, as well as formal training for alliance managers, alliance summits, and so forth. Partner-specific common coordination knowledge that enhances cooperation efficiency increases with the number and duration of projects in which the partners have cooperated. For example, the partners set up standardized procedures so that they do not have to negotiate how to design cooperation processes anew for each project. Partners that have accumulated a substantial amount of coordination knowledge would tend to economize on the exchange of content knowledge. Alliance managers who are specialized in the management of specific partnerships would also be likely to quickly provide the necessary resources and specialists from their companies for a beginning project.

As discussed already, some overlap of content knowledge may be useful for coordination between specialists in spite of TOL. The present study assumes that the volume of overlapping content knowledge depends on the extent of tacit knowledge involved in the coordination. The following examples illustrate the plausibility of this assumption. Chefs who are specialized in meat usually know quite a lot about preparing sauces. They can therefore communicate effectively with their colleagues if a different taste of a sauce is called for to make the meal harmonious. However, designers of car bodies need not know a
great deal about engine design to be able to cooperate effectively with engine designers. A language for the description of required inputs and outputs should, as shown here already, be sufficient. Referring to these examples, difficulties in defining intended inputs or outputs that are dependent on the extent of tacit knowledge apparently determine the extent to which overlapping specialized knowledge is required. Meat chefs who know how to produce sauces are in a better position than chefs who do not possess this knowledge to specify required changes in tastes in terms of inputs or variations in the production process.

Shared understanding of the intended product is also supposed to be helpful (Grant, 1996b). When cooperating, a car body designer and an engine designer need less verbal communication if they share the same understanding or vision of the intended car. What is needed is a big picture that allows the members working on the different modules or subsystems to fit their part into a larger frame. Or, as Dougherty (1992, p. 182) describes, “the systems of meaning produce an ‘intrinsic harmony’ for the thought world, so ideas that do not fit may be reconfigured or rejected outright.” Shared understanding is presumably more important for modular innovations than for architectural ones for which a description of the desired functions is not overly difficult to generate.

This reasoning leads to the last proposition.

**P5:** Common knowledge effectively supports cooperation efforts in strategic alliances for architectural innovation. It should include coordination knowledge and, dependent on the share of tacit knowledge involved, content knowledge that overlaps between partners.

**Supporting and Extending the TOL Concept through Case Studies**

**Methods and Sample**

As Eisenhardt (1989, p. 535) points out, case studies can be used to accomplish various purposes, namely “to provide description, test theory, or generate theory.” The case-study approach was applied in the present study to find empirical support for the existence of the elements of the TOL concept and their interactions and also to enrich this concept by collecting information about ways these elements are applied. The literature on case studies stresses the importance of “prior development of theoretical propositions to guide data collection and analysis” (Yin, 2003, p. 14). Without such a research focus “it is easy to become overwhelmed by the volume of data” (Eisenhardt, 1989, p. 536). The specific design of our case study approach is the embedded single case—that is, several units of analysis are embedded in one context, or one organization (Yin, 2003, p. 14), which allows for controlling external, or organizational, variables and for concentrating on variables that characterize the phenomenon in question. In the present case, by choosing architectural innovations in alliances in which one partner is kept constant, insight can be gained into possible influences on learning modes that are linked to different kinds of projects.

Evidence for the existence of the TOL concept and its mechanisms has already been collected in a multiple-case study that comprised two cases on the redesign of organizational rules: in a bank and in a manufacturing company (Kieser and Koch, 2002; Koch, 2004).

On the basis of this design, four product innovation projects were studied in strategic alliances of SAP, one of the leading producers of standardized business software, with other partners (for a detailed project description and methodological discussion see Grunwald, 2003). The “sampling” of the cases followed the principle of “comparable case selection” (Goetz and LeCompte, 1984, p. 64) or “literal replication logic” (Yin, 2003, p. 47), which means that the cases are supposed to be similar with respect to the focal attributes by applying the following criteria: (1) one or several teams staffed by the cooperating parent organizations; (2) substantial knowledge input from each partner; and (3) completed architectural innovation projects (allowing for the retrospective reconstruction of recombination processes of knowledge). Architectural innovations were defined as innovations for which known technologies are integrated, leaving their core designs intact. In addition, the projects should vary with regard to SAP partner type—hardware technology, services, and software—to test whether the findings are stable with regard to the partners’ technologies. After all these criteria had been applied, only a few projects remained. Among these the most recent ones were selected.

The first project is a cooperation between SAP and MSG Systems AG, a medium-sized German consultancy and software development company specialized in insurance. This project included the development of a commissions calculation system for the insurance sales force. The project took two and a half years.
Temporarily, around 20 experts were involved working together in shared offices at the SAP headquarters in Walldorf near Heidelberg, Germany. The SAP specialists’ contribution consisted in software and standardization know-how, whereas the main MSG experts’ input consisted of SAP-R/3 consulting competence and knowledge about the insurance industry. There was a moderate degree of experience between these partners due to previous common projects.

The second project concerns a collaboration between SAP and Elecktron AG, a major German manufacturer of electronics and communication network equipment; in contrast to the other companies of this sample, the Elecktron’s management preferred their company to remain anonymous. It aimed at combining the SAP business software for customer relationship management (CRM) with an Elecktron communication engine for call centers into a new product allowing the intelligent acceptance and routing of customers’ inquiries (i.e., business routing in call centers). During the two-year development time, the team members (15–25 depending on the phase of the project) worked at three sites: Walldorf near Heidelberg, the SAP headquarter; Hamburg; and Calgary. In this project they combined the knowledge of SAP in the area of software for CRM with Elecktron’s know-how about communication systems for call centers. In this alliance, alliance managers were found to be in charge of coordinating activities of these two partners, indicating a high degree of alliance-specific coordination knowledge. This partnership looked back on a number of previous common projects.

In the third project, Asian language support, SAP cooperated with IBM to create software that makes the combination of SAP systems and IBM components—hardware, operating, and database systems—applicable for customers in Asian countries. The team comprised 35–45 members, depending on the phase of the project, and was located in Walldorf and Rochester. Most of the team members worked together in the shared offices at SAP’s headquarters. SAP brought in its software, SAP-kernel, and the connected know-how; IBM brought its experience with the computer system AS/400, its associated operating system OS/400, and the database DB/2. Within the sample this alliance had been working on common projects since 1976 and had thus accumulated the highest degree of alliance coordination knowledge.

The fourth project, a cooperation between SAP and Dun and Bradstreet Corp. (D&B), involved the integration of data from the D&B worldwide business information databases into SAP’s Business Information Warehouse. The eight team members working at two sites of the SAP headquarters and High Wycombe, United Kingdom, took one year to finish the project. D&B contributed its extensive company data and its business know-how regarding financial reports; SAP’s contribution consisted of software and the associated development knowledge. Coordination between the partners was supported by alliance managers who were in charge as alliance managers for other alliances as well. Thus, existing coordination knowledge can be assessed as medium. All the projects of the sample are software engineering projects. This means that tacitness of knowledge for all projects is low. Table 1 gives an overview of the analyzed projects.

In three of the four projects—Elecktron–SAP, IBM–SAP, and D&B–SAP—the subteams were based at different locations. Since in each case the resulting output is software that can simply be stored on CDs, the question regarding the final products’ production site becomes secondary to the issue of appropriation and revenue sharing. Although in the MSG–SAP project the application software is sold as proprietary to SAP, there are joint sales agreements in the other three cases; that is, both sides have integrated the developed interface into their software and are marketing and selling it together with the partner.

In this context, a detailed discussion of the peculiarities of the different cases with respect to the TOL concept is not possible. Furthermore, the main objective of this research was to find empirical evidence for the model across the different cases.

The semistructured questionnaire used for the interviews with members of both partners in each of the four projects is found in the Appendix. The first author conducted 31 interviews between December 2000 and April 2002: 8 in project MSG–SAP; 12 in project Elecktron–SAP; 7 in project IBM–SAP; and four in project D&B–SAP. Three of the interviews were conducted in English with Anglophonic team members. The questionnaire aimed for an ex post reconstruction of the process of knowledge recombination processes in the different stages of the research projects. The interviews lasted between 30 minutes and two hours, with an average duration of 80 minutes. Twenty-eight interviews were recorded and completely transcribed. In the three cases in which the interviewees did not allow recording, notes were taken during the interview and were completed immediately following the interview. In addition to the interviews, various documents
<table>
<thead>
<tr>
<th>Project</th>
<th>SAP Knowledge</th>
<th>Partner Knowledge</th>
<th>Output</th>
<th>Type of Innovation (Architectural vs. Modular)</th>
<th>Prior Cooperation Experience of Alliance&lt;sup&gt;a&lt;/sup&gt;</th>
<th>Strength of Alliance Coordination Knowledge&lt;sup&gt;b&lt;/sup&gt;</th>
<th>Degree of Tacitness of Combined Knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP-MSG</td>
<td>Software and know-how on standardization</td>
<td>Know-how on insurance companies</td>
<td>Application software for insurance companies</td>
<td>Architectural</td>
<td>Medium (cooperation since 1980s; no prior cooperation in R&amp;D, only regarding consulting)</td>
<td>Medium (no alliance manager, but steering board installed more than three years before analyzed project started)</td>
<td>Low (software engineering)</td>
</tr>
<tr>
<td>SAP-Elektron</td>
<td>Software for customer relationship management</td>
<td>Communication engine for call centers</td>
<td>Computer telephony integration (business routing in call centers)</td>
<td>Architectural</td>
<td>Medium (alliance founded two years before analyzed project started; limited to single projects)</td>
<td>High (dedicated full-time alliance managers)</td>
<td>Low (software engineering)</td>
</tr>
<tr>
<td>SAP-IBM</td>
<td>SAP-Kernel (especially R/3)</td>
<td>Hardware (AS/400), operating system (OS/400), and database system (DB/2)</td>
<td>Asian language support for SAP application software on IBM hardware</td>
<td>Architectural</td>
<td>High (alliance since 1976; systematic cooperation)</td>
<td>High (dedicated full-time alliance managers)</td>
<td>Low (software engineering)</td>
</tr>
<tr>
<td>SAP-D&amp;B</td>
<td>Business information warehouse (BW)</td>
<td>Global database, reporting knowledge</td>
<td>Integration of D&amp;B data in SAP-BW (Vendor/Customer Analytics)</td>
<td>Architectural</td>
<td>Medium (alliance since mid 1990s; only single projects)</td>
<td>Medium (Part-time alliance managers)</td>
<td>Low (software engineering)</td>
</tr>
</tbody>
</table>

<sup>a</sup> Refers to the moment when the analyzed cooperation projects started. There are three degrees: (1) high, alliance older than three years; systematic cooperation in more than one dimension (e.g., R&D, sales, marketing) and in multiple projects; (2) medium: alliance between one and three years old; limited to single sporadic projects; and (3) low, alliance younger than one year; not more than one project.

<sup>b</sup> Measured by its institutionalization with a dedicated alliance manager or an equivalent organizational structure (e.g., steering board): (1) high, dedicated alliance manager with full-time job; (2) medium, dedicated alliance manager with part-time job; and (3) low, no dedicated alliance manager.
including minutes from meetings between the two partners’ project members or descriptions of products or components were analyzed.

The data interpretation was inspired by Miles and Huberman’s (1994) interactive model, which recommends three steps: data reduction, data display, and drawing/verifying conclusions. Using the Atlas.ti tool, a code scheme with 36 codes was applied, which had been developed at the beginning of the data analysis and was modified during its course (Araujo, 1995; Coffey and Atkinson, 1996).

Findings

This section provides the results of the study. All quotations given from non-English-speaking interviewees were translated into English by the authors. The quotations are identified at the end in parentheses with the following information: project, number of interviewee, number of quotation.

The reduction of the need for interorganizational and interspecialist learning through modularization. No support was found for the dominant view that product innovation in strategic alliances requires extensive knowledge sharing between the partners. On the contrary, the strategies identified in this study’s cases definitely aim at limiting the exchange of knowledge. The interviewees emphasized the efficient integration of knowledge bases as a process that is clearly distinct from in-depth interpersonal or interorganizational learning. Learning from each other would have been, as they stressed repeatedly, a much too time-consuming process. The basic exercise is to put competencies together:

That each one has its core areas and core competencies had not been doubted by the respective others . . . . The point was, how to get the two together. The point was not to question the competencies of the other or to learn them somehow, but simply how to integrate the respective competencies. (MSG-SAP, 3, 16)

The strategies applied were based on the principle that each group should stick to its respective specialization as much as possible. This goal was achieved by carefully designing the modules the different groups had to design and the interfaces between them. The interviewees pointed out that, based on a modularized architecture, they usually tried to arrive at a mode of cooperation that would allow both partners to work relatively independently of each other:

In the design phase one reaches a good separation of tasks if the interface is sufficiently clear. That’s an enormously important phase in such a project between two or more partners; one has to clarify which components belong to which part of the final product: what is SAP performance, what is Elecktron performance and how is the interface between these components defined syntactically and semantically. (Elecktron–SAP, 6, 68)

In spite of being cooperation partners, it makes sense to separate the tasks absolutely clearly into components or modules that are independent of each other. (Elecktron/SAP, 10, 61)

In all projects, subgroups were organized around modules or components. Due to the division of work between groups, which tended to concentrate communication within subgroups, and to minimize communication between them, content knowledge—as opposed to coordination knowledge—was exchanged between these groups in a highly selective manner and only when required by emerging problems. At the beginning of this kind of project, the intended product’s architecture is developed, which involves designing modules and their interfaces. This process includes a discussion of the modules’ functionalities and the inputs they require from other modules.

However, this initial discussion does not go into great detail. It suffices that the project team members get a first impression of the functionalities of the modules they are assigned to design. After the completion of the project architecture, the various module teams can work as loosely coupled subsystems (Colville, Waterman, and Weick, 1999). In some cases “loose coupling” entails linking across large geographical distances. In the Elecktron–SAP project, for example, the groups were working in different time zones on different continents. This study’s data show that there was a very low degree of interaction between the groups. Over several weeks only the project leaders communicated during prescheduled weekly link-ups.

Storage media. In all four projects the coordination processes centered around descriptions and concepts of modules and the projected product. Usually these were stored electronically so that all members of the project team had easy access to them. With the
evolution of this type of project, the concept descriptions become more accurate and elaborate. According to their degree of detail and refinement, these concepts are called first draft, second draft, and so on up to the final draft. The first concepts of the product and its parts are rough descriptions of projected functionalities and required inputs and outputs. Nevertheless, these rough descriptions provide orientation so that the subgroups can start working. In their efforts to construct or program their respective modules in accordance with the overall concept, the subgroups gain insights and accumulate experience that is reflected in revised, more detailed concepts, including revised specifications for outputs and inputs needed from other modules.

Transactive memory systems. As pointed out previously, in projects for architectural innovations specialists must be able to quickly localize required knowledge they lack. A vice president of SAP expressed this requirement in the following way:

Efficiently connect those who know with those who need to know. (Elecktron–SAP, 4, 63)

Transactive memory systems were present in different forms in all four projects. Project team members do not have to acquire all knowledge needed in the course of a project or engage in a lengthy search for specific knowledge that they lack, if experts with this knowledge can be identified through a directory of specialists—or a team member with such knowledge. One of the interviewees labeled this mechanism Ostfriesenprinzip (i.e., the principle of the Ostfriesen—the Ostfriesen are people from Northwest Germany who are ridiculed as being especially dull witted).

The best approach is always: one knows, by chance, somebody, from earlier times, who is able to contribute something or who knows somebody. Like the old principle of the Ostfriesen: one knows somebody who knows somebody. That's what works best within SAP. (MSG-SAP, QU 2:66)

Directory knowledge is unevenly distributed among the members of a group. Some are “directory specialists” (Wegner, 1995). In the MSG–SAP and IBM–SAP projects the project leaders who had accumulated a lot of experience and built up extensive personal networks during many years of work for their companies took over the role of directory specialists. In the Elecktron–SAP and the DandB–SAP projects the alliance or partner managers distinguished themselves as members with broad directory knowledge. In a certain way, they acted as hubs for localizing and contacting specialists in their respective home organizations. In response to the question, “If you needed someone [an expert] from the SAP side, how would you proceed?” one interviewee answered,

I just would contact DG [the alliance manager] and tell him, “We need contacts for . . . the themes 1, 2, and 3.” And then it is the task of the partner management on the side of SAP to identify these contacts and to brief them in such a way that they can become useful for the project. (Elecktron–SAP, 7, 41)

If no team member was able to retrieve a needed specialist from his or her directory, the team at least managed to identify somebody “who was supposed to be able to help” (Elecktron-SAP, 3, 67). If such a person could not be identified, it was, in most cases, obvious to which department or area the needed specialist was likely to belong. In these cases the department head or somebody else in a higher hierarchical position was contacted. One of the basic functions of a hierarchy is to bundle directory knowledge—knowledge about who knows what on lower levels—on higher levels (Schulz, 2001). This specialist identification function of the hierarchy is often supplemented by specific databases or electronic directories (i.e., electronic yellow pages): Fields of specialized knowledge are linked with names of organizational members. However, this study’s interviewees emphasized that in most cases directories based on personal networks are superior to electronic ones. This can be explained with two properties: Information drawn from a personal network is richer in context, and updating it is easier so that the information will generally be more up to date and trustworthy than in anonymous electronic files.

Recombination of knowledge through prototyping. Empirical evidence was furthermore found for the subconcept in which is described how repeated discussions of concepts between groups and, in later stages, simulations and real tests of linked modules lead to modifications of modules until a satisfactory final concept is reached or, ultimately, the product is finished. In response to the question, “How did you solve problems for which knowledge from both sides was required?” one interviewee answered,
This was achieved through communication, but this occurs in iterations. Things are misunderstood, are done wrongly. Then, in a second iteration, one needs to improve the whole thing. And, sometimes, one needs several iterations. (SAP–Elecktron, 5, 40).

The same interviewee, in response to the question, “Such an iteration is based on software?” answered,

Yes, one believes that one agrees, implements, until one discovers that something is going wrong. Then one asks, there must be a bug somewhere in your program.

And another interviewee, in response to this question, answered,

That’s one of those trade-offs of a formal development process versus time to market. That’s a difficult judgment, a trade-off between spending a lot of time doing high-level design and reviews versus just going ahead, constructing prototypes, and trying things, which is more the SAP approach. We tended to follow the SAP approach. (SAP–IBM, 4, 47)

Prototyping does not require time-consuming processes of interdepartmental or interorganizational learning. On the contrary, by speeding up the development process on the microlevel of the individual developer (i.e., output-oriented prototyping instead of abstract high-level design; see quote from SAP–IBM, 4, 47) it considerably reduces time to market. In a certain way, the prototype learns by being exposed to critical evaluations by specialists of other domains and then being revised according to their feedback. Later on in the process, learning is based on tests and real prototyping. Revisions of concepts and prototypes are implemented without the respective group having to thoroughly understand why these modifications are necessary. For example, if the communication engineers from Elecktron said, “We need to adjust line 453 in the kernel of our call-center software,” the SAP software engineers taking care of CRM software would not insist on an in-depth explanation of this request but would trust their colleagues from the partner organization. In essence, they modify their concepts and, later, prototypes because other groups working on other modules want them to do so on the basis of plausible arguments—as opposed to comprehensive explanations.

The project teams in these cases used several technical devices for presenting and evaluating concepts and prototypes. For the Canada-based members of the Elecktron project, remote access to the SAP systems in Germany was established so that they could test the functionality of their prototypes through simulations. Remote access was also implemented in the D&B–SAP project. Apart from remote access to SAP systems, Microsoft NetMeeting, an Internet-based system for distance conferencing, was used. With these technical means, it is, for example, possible for the developers residing in different locations to study the same model on their screens and at the same time to discuss changes and to implement them by applying a higher-order programming language via keyboard. With the help of these devices, the prototyping mechanism also works across long geographical distances.

Prototyping was implemented by getting not only specialists from other project groups to provide feedback but also external specialists, either from pilot customers or other organizations. These external evaluators are supposed to discover problems or inconsistencies in the processes that have been overlooked by the members of the project team. External evaluators get involved predominantly in two phases: before programming starts and before the final product is launched. The interesting point is that this external prototyping follows the same principles as the internal prototyping: People from outside the core team evaluate concepts and prototypes, and their feedback is used for improvement.

The role of common knowledge. A problem mentioned as emerging frequently at the beginning of projects was that the partner’s specialists used organizational and specialist jargon. However, this problem was not perceived as being especially serious. After some time the project team members from the partners understood each other. Some project teams solved the problem by compiling glossaries. In other projects, members from the partners learned each other’s vocabulary through discussions in which misunderstandings were clarified. Moreover, the object-oriented method applied by most software engineers in the projects studied, which can be categorized as an artificial language or coding system, proved to be highly supportive for the communication between specialists of different domains.

Regarding the commonality of content knowledge, it was found that the members do in fact exchange content knowledge across the borders of their respective parent organizations, especially in the early project phases. However, the bulk of their specialized
content knowledge is directly applied to the design and engineering of their respective modules' concepts and prototypes. The inclination to share specific content knowledge with project members from the partner is extremely limited. Common content knowledge is built up in a highly selective manner—that is, only when emerging problems require intensive coordination, which has only rarely been the case. As pointed out already in the example of the chefs and the car manufacturing engineers, as long as module inputs and outputs are not too difficult to operationalize (i.e., low degree of tacitness of combined knowledge)—as is the case with all the software engineering projects for this study—very little exchange of content knowledge is required. It usually suffices to understand the general business logic (i.e., how the product functions):

It is essential, that I do not learn bits and bytes from the SAP side (i.e., what the data structure [and the source code] looks like) but that I understand the business logic, which they can create with their customer-relationship mask. (SAP–Elecktron, 10, 38)

In addition to these findings supporting the study’s assumptions regarding common content knowledge, the interviews produced a number of explorative results. In particular, it was found that the extent to which content knowledge is exchanged between groups working on different modules, which also includes groups from the partners working on different modules, depends on the following factors:

(1) Previously existing shared knowledge: If members from two groups whose modules are supposed to interact with each other in the final product already possess overlapping content knowledge, such as because of previous cooperation experience, the extent of content knowledge learning between these groups is reduced.

(2) Project phase: Ceteris paribus, content knowledge exchange between groups is more intensive in the early project phases.

(3) The group members’ role: Official project team leaders are responsible for most of the exchange of information and knowledge between subgroups. If the project team leaders, acting as hubs, understand the requirements for the specific modules, communication can be organized much more efficiently compared to bilateral communication between the subgroups. However, this does not mean that the project leaders in this study’s cases acquired and passed on in-depth specialized knowledge from the different subgroups. On the contrary, the project leaders stressed that they did not have to understand the different subgroups’ modules “down to the last bit and byte” (Elecktron–SAP, 10, 38):

I do not see myself as a designer of solutions, since I do not believe that—though I was a software engineer myself for some time—I am always the best programmer in each field. That’s nonsense; I said farewell to this view a long time ago. I see my role as involving the definition of requirements, of the rough guidelines, and thinking about design problems only on the highest level. (IBM–SAP, 7, 53)

Similarly, one project leader emphasized that none of us was a mastermind understanding everything. (IBM–SAP, 4, 60)

(4) The existence of alliance or partner managers: In three of the four projects there were partner or alliance managers in addition to the project team leaders (Kale, Dyer, and Harbir, 2001). When SAP is expecting a certain number of further strategic alliances with partners such as IBM, Microsoft, HP, or Accenture, typically a partner or alliance manager is institutionalized (i.e., a manager who acts as a kind of coordinator and troubleshooter for all projects within this partnership). If one assumes that the project coordination problems within a certain partnership are similar in some respects, such a coordinator can accumulate specific alliance coordination knowledge that generally supports cooperation between these partners. The team leaders in the present study emphasized that learning from members of the partner organization concentrated on gaining a basic understanding of the know-how brought in from the respective partner but that this understanding was neither in-depth nor encompassing. This finding is not surprising as in all projects there was at least a medium degree of cooperation experience between the partners. As pointed out previously, the existence of coordination knowledge reduces mutual content knowledge transfer.

Similarly, the importance of shared systems of meaning, according to this study’s findings, should not be overstated. In principle, the specialists stuck to their specific approaches and waited for the results of the prototyping process to uncover inconsistencies that might have been caused by differences in basic understandings among the specialist groups involved. The following quotation illustrates the managers’
modest expectations with respect to the development of a common understanding:

Because of the organizational culture, the origin of people [project members], there was, initially, a different basic understanding and a different thinking. Each person came from a different world of thought. And this had to be integrated. The Elecktron people came from the technical side, from the technical world, and we [SAP] from the implementation–development side. We are pretty close to the users, as far as software development is concerned. For us, management science and management processes are much more in the foreground. The different kinds of thinking have to come together. (SAP–Elecktron, 5, 37)

There were no indications of attempts to establish shared organizational maps (Argyris and Schön, 1978, p. 17) or common perspectives (Nonaka, 1994, p. 24). It is sufficient to understand how and why the members of the partner organizations think differently. The members tried to arrive at a rough common interpretation of the specific problems that had to be solved. This common-problem understanding focused on the functionalities of the whole system and its subsystems and did not extend to a system of meaning or a common view on problems in general.

What both sides had to understand was the master plan of the software that we were supposed to develop. (MSG–SAP, 3, 100)

The required common understanding of problems, as defined by the interviewees, is a rough idea about how the components or modules that are developed by each partner should interact. The partners’ different systems of meaning were acknowledged and taken into consideration when members of one partner interpreted concepts of the other. But no efforts could be identified to merge these different meaning systems into a new and all-encompassing one. The following quotations illustrate the kind of common understanding that was needed:

One needs a common understanding that has to be accomplished in a technical sense. (Elecktron–SAP, 9, 17)

Across all subteams each one knew where his contribution had to fit in, knew the master plan, so to speak. (IBM–SAP, 7, 73)

This common understanding of problems was created in the project’s initial phase. However, it was not shared to the same degree by all team members. Some team members—especially the team leaders and alliance managers—engaged more than others in efforts to establish it. Also, there were indications for the importance of coordination knowledge.

Revisions of one module frequently require adaptations of other modules and, therefore, coordination between subgroups. Common coordination knowledge helps to make this cooperation efficient. For example, the group members have learned to interpret highly abstract flowcharts and other forms of symbolic representations of the subsystems’ functionalities and are thus able to grasp the essential coordination problems much faster than individuals who do not possess these skills. Artificial languages of this kind work like musical notations and the conductor’s language of hand signals and movements that enable orchestra members to efficiently communicate and interact with one another. Communication across groups of specialists is different from communicating within these groups.

I think that an important point or the most important point is that one tries to understand the other guy and that one tries to formulate what one intends to communicate in such a way that the other guy can understand it. A methodology is important here—that means a methodology that allows a reconstruction and which is based on rules, a common language, ultimately. One can never exclude diverging interpretations, but ultimately one discusses around a box on which there is written, “This is a commission contract.” Then one discusses properties of such a thing, and one speaks a common language. This takes time, but then it works relatively well. (MSG–SAP, 4, 63)

Discussion and Conclusion

Organizational members’ absorptive capacity is limited; therefore, specialization is an indispensable organizational principle. With respect to this principle and in contrast to the dominant view, this study argues that specialists from the partners of strategic new product alliances should not attempt to intensify but rather, at least in architectural innovations, should attempt to keep cross-learning at a moderate level, since it is a costly mode of knowledge integration. This goal can be achieved by applying four integrated mechanisms: (1) modularization; (2) storage
of knowledge in media such as concepts and prototypes; (3) transactive memory; and (4) prototyping of concepts, of models of modules, real modules, and completed integrated real modules. A qualitative analysis of four SAP innovative projects with different alliance partners provides support for this study’s TOL concept. The interviews show that designers consciously apply these mechanisms because they are aware of their advantages over other forms of knowledge integration. The mechanisms of the TOL concept help to reduce exchange of content knowledge in strategic alliances for architectural innovations. Also in the empirical material a number of additions and necessary modifications of the original concept were identified. For example, a number of elements were specified whose presence in common knowledge contributes to ease of cooperation. In addition, factors were identified that impinge on the necessary extent of common content knowledge. And these findings point to the relevance of accumulated coordination knowledge in long-standing partnerships. The interviewees also agreed that directories in the form of personal networks are more useful than electronic yellow pages. All these aspects were not contained in the original concept.

To sum up, interspecialist and interorganizational learning is not as important as most researchers of interorganizational cooperation want us to believe. An integration of existing technologies can be achieved through the TOL concept. This concept corresponds to a fundamental organizational requirement as stated by Demsetz (1991, p. 172): “There must be a low-cost method of communicating between specialists and a large number of persons who are non-specialists or who are specialists in other fields.”

**Limitations of the Empirical Study**

A first limitation is that the study concentrated on technical issues of architectural innovations and did not consider other functions of the product development process, such as marketing. The study assumes—and some information from the cases supports this assumption—that the process is similar: Marketing concepts are discussed among different experts of the partners who are familiar with the different markets of relevance. Feedback leads to modifications of the marketing concept. These are implemented without an in-depth exchange of content knowledge regarding marketing solutions. There are additional limitations. Radical innovations, as opposed to architectural innovations, may require different approaches to organizational learning. In particular, it can be assumed that a shared understanding of the reality that will result from the intended innovation might be of greater importance for these innovations (Garud and Rappa, 1994). However, innovations commonly categorized as radical, such as the steam engine, are on closer inspection essentially also recombinations of established technologies (Rosenberg, 1976). Therefore, the range of innovation processes that can be analyzed with this study’s concept might turn out broader than one might at first sight suppose.

Moreover, all the projects in the study’s sample are, in essence, software engineering projects. It is conceivable that more physical technologies require different approaches for specialization and coordination. Technologies that involve a high degree of tacit knowledge, which usually is the case with physical technologies, may require different approaches for reducing interpartner learning. However, we are convinced that software engineering is only an extreme case of recombing existing technical knowledge—a case that allows for analyzing problems of recombin- ing technological knowledge in their pure forms. Necessary modifications of the TOL concept are easier to identify once a relatively uncomplicated case has been identified. The TOL concept has also found support in case studies on redesigning organizational rules (Kieser and Koch, 2002; Koch, 2004), that is, for innovations that are radically different from architectural software innovations.

Another limitation is that the study concentrated on the development phase of product innovation efforts within strategic alliances. As several interviewees said, agreement on sales arrangements is often more susceptible to opportunistic behavior of partners (Combs and Ketchen, 1999) than cooperating in the development of a joint product innovation. Also, the innovations analyzed were embedded in existing relationships between the partner companies, in which trust had accumulated. And, finally, all cases in the sample had one big company—SAP—as a dominant partner. Perhaps other partners develop patterns of cooperation that deviate from those identified in this study.

For a long time scholars in organization theory have been discussing approaches to increasing an organization’s capacity for learning, especially for learning across different domains of specialization and
organizations. They came up with ideas like improving the conditions of organizational learning through the creation of a common social reality (Nonaka, 1994) and through the implementation of the roles of translators (Leonard-Barton, 1998) and brokers (Brown and Duguid, 2002). Even a despecializing of specialists has been proposed (Postrel, 2002). This study suggests that efforts to identify approaches and mechanisms for a reduction of the need for organizational learning may turn out at least as promising.

Implications for Practice

The TOL concept identifies four mechanisms—modularization, knowledge storing in artefacts, transactional memory, and prototyping—that significantly reduce time to market. Companies should find out—possibly on the basis of benchmark studies—whether these mechanisms are present and exploited to the fullest degree. In such a survey they should pay special attention to the following points.

Technologies of other companies offer possibilities for architectural innovations. Therefore, companies should take into consideration institutionalizing scanning other companies for technologies that could be recombined with their own ones (Bozdogan et al., 1998; Cebon, Hauptman, and Shekhar, 2002; Henderson and Clark, 1990; Khanna, Gulati, and Nohria, 1998). Our discussion partners within SAP reported that in bigger software companies, specialists systematically search for white spots—that is, software solutions that are not yet developed but could have potential for groups of customers. Once identified, white spots trigger a decision whether to develop the solution oneself, to buy it, or to develop it with a partner. If the decision is in favor of a partner, potential partners have to be checked and consulted.

In alliances of this sort, as this study’s cases suggest, the role of alliance managers should be established. The alliance managers take care that standards of knowledge on how to cooperate spread over all projects of an alliance and—in cooperation with other alliance managers—even across all the alliances of a company. Of course, partners might pursue the same strategy so that compromises over changing partners’ standardization strategies have to be made.

It seems especially important to develop and apply computer-based tools for the effective communication between groups of specialists, especially tools for presenting concepts and for prototyping (Khanna, Gulati, and Nohria, 1998; Lorenzoni and Baden-Fuller, 1994; Thomke, 1998). In other words, effective common coordination knowledge reduces the need to share content knowledge.

In spite of many modules being predefined by the technologies that are supposed to be combined, architectural options still exist, which means that the identification of effective modularization strategies is of crucial importance and deserves particular attention of practitioners and management researchers (Ethiraj and Levinthal, 2004; Mihm, Loch, and Huchzermeier, 2003).

The interviewees did not consider SAP’s system of electronic yellow pages to be effective, though it was, as imagined, technically up to date. The same applied to Elecktron’s company-wide yellow pages. Interestingly, in these two companies, the yellow pages provided on a corporate level are only used by the few people who deal with knowledge management. In smaller Elecktron units, such as the German sales and consulting force, skill databases are existent but are only used by those who decide on the employees’ deployment for specific projects. Studies show generally that many electronic yellow pages do not live up to their promises as (1) they only cover in-house knowledge holders; (2) they are incomplete because a complete system requires a lot of effort for inputting relevant information whereas the incentives for those who provide inputs are highly uncertain; (3) experts tend to not flag knowledge that has not been of importance in the past; and (4) the terms used to index knowledge are often not the ones used in searching this knowledge. Efforts to overcome these difficulties are under way but have not yet resulted in satisfying solutions (Birkinshaw and Sheehan, 2002; Borghoff and Pareschi, 1998; Dooley, Cormans, and McPhee, 2002; Hustad and Munkvold, 2005; Persaud, Kumar, and Kumar, 2001). That efforts to identify experts should include external ones is pointed out, among others, by Ritter and Gemünden (2003) and Walter (2000).

References


Appendix. Interview Questionnaire

(1) Introductory Questions

(1) What kind of collaboration does your company have with the partner company? What activities take place within the cooperation (e.g., common projects)? (Questions in parentheses are “prompts,” that is, they are asked only if the interviewee does not know an immediate answer).
(2) Could you please explain what this project was about?
(3) What were your tasks in the project?
(4a) How long did you work on the project?
(4b) When was the project launched?
(4c) For how long has there already been a collaboration between your company and the partner?
(5a) What kind of know-how (most important areas) did your company contribute to the project?
(5b) What kind of know-how (most important areas) did the partner company contribute to the project?

(2) Initializing Phase of the Project

Did you work on the project right from the beginning? 
(If no, proceed to question block 3; if yes, please answer the following questions.)
(1a) How did the project start?
(1b) With how many people on your side and the partner company’s side?
(1c) How was it triggered?
(2) How did it go on the following: system specification, desiderata, technical descriptions, milestones, roadmaps?
(3) Why were you selected or placed on the project (e.g., your own initiative and interests, special know-how, somebody invited you to join the project)?

(3) Development Phase of the Project

(A) Learning Strategies and Integration Mechanisms for Dispersed Knowledge
Introduction: The following questions deal with the topic of approaches for developing a product (i.e., learning strategies). In this project, the employees of both your company and the partner were specialized in different areas. The team had to use these different inputs and create a single, joint output. Generally speaking, our central questions are as follows:

(1) What was your approach to integrating the know-how among the different experts from the two companies? (Did you employ standardized procedures? Did you make use of demo versions or prototypes?)
(2) How were the different inputs coordinated when several employees from both companies were working simultaneously on the same component? (Did you use technical instruments?)

(B) Search for and Localization of Knowledge to Be Recombined

(1) How did you identify people with expert knowledge who did not belong to your team’s staff and (a) were from your company? (b) were from your partner? (Was there a dedicated contact person for such cases?) (c) were external experts? Did you ask some employees more often than others to localize experts?
(2) How did you get to know who knew what? (Did you have commonly accessible expert directories like yellow pages or the Intranet? Or was this knowledge just in your head?)
(3) If you had a technical problem in the project for which you needed more information, how did you acquire that knowledge? (a) Did you call on a specific person you thought could help you; did you approach employees from your partner as openly as colleagues from your own company? or (b) did you try to acquire that knowledge on your own, for example, through books, articles, Internet, or Intranet?
(C) Common Knowledge and Shared Understanding

(1) How important was it for you to understand the contributions the partner company employees provided for the project? (a) How much of your own technical knowledge did you need for that? (b) Did you learn a lot from the partner company employees?
(2) How did you share relevant knowledge in the project (e.g., Intranet, e-mail newsletter, trainings)?
(3) Did you ever have the situation in the project that team members from your side did not understand the know-how brought in from the partner (or vice versa)? Was this a problem? If yes, how was this problem solved?
(4) Did you have the impression that the project team developed its own identity? An identity encompassing the different corporate cultures from your company and the partner? (Were there unifying stories, shared understanding between project members or common spare time activities?)

(D) Communication and Codification, Cooperation Knowledge

(1) Within the project team, did you experience situations with communication problems (e.g., because of words, abbreviations, terms, or expressions only known to members of one company and therefore incomprehensible for the counterpart)? (a) If yes, what kind of communication problems? (b) Were common wordings and labels or a “common language” helpful in such cases? (c) Did you standardize expressions which you as team members could then use?
(2) Were there any communication problems between team members with a more business-oriented background and those with an engineering or software developer background? Particularly for software engineers, was there a unifying common background?
If not yet answered in question A1:
(3) In cooperating with the partner company, did you make use of standardized tools in the project when you had a joint task to carry out (i.e., tools to track the project progress such as the standard software MS Project or self-developed software)? Did you make use of any other written documents or written tools (e.g., memos, handbooks, blueprints, MS-Excel spreadsheets, decision support systems, commonly used MS Word documents)?

(E) Other Questions

(1) Had you already gathered experience in collaboration projects with other partners? (If no, continue with question E3; if yes, please answer the following questions.)
(a) Did this experience prove helpful to your tasks in the project (e.g., routines of how to integrate knowledge)?
(b) Could you give some examples, please?
(2) In your opinion, what are the most important factors in managing a collaboration project?
(3) How do you judge the openness of knowledge sharing in the project? (a) Was there information or knowledge which was not shared? (If yes, what kind of knowledge?) (b) Would you share your knowledge with a partner team member in the same way as with a colleague from your own company?

(4) Finalizing Phase of the Project

(1) What were the results of the project?
(2) Upon completion of the project, was a discussion ever held about positive and negative experiences? If so, in what form (e.g., formal evaluation)?

(5) Subjective Evaluation of the Project’s Productivity

(1) What did you like? What could have been improved with regard to (a) the way the knowledge from your company and the partner company was newly combined? (b) the output and the results of the project?
(2) Please give reasons for your answer.
(6) Closing Questions

(1a) What kind of educational experience do you have (e.g., business background, software developer, engineer)?

(1b) What professional experience do you have (i.e., previous jobs or projects)?

(2) Are there any other people I should interview for our study (e.g., from your company, from the partner company)?

(3) Do you have any documents that could be helpful to the purpose of our study?

(4) Is there something important with regard to the development work in the project about which we have not talked so far?