SARA
A Software Reuse Architecture for Building Expert WorkFlow Systems

Einar Dehli, Benedikte Harstad Kallåk, Thomas Bech Pettersen, Jan Erik Ressem, Are Sørli, Geir Waagbø

Computas AS
P.O. Box 444, N-1301 Sandvika, Norway
Phone: +47 67541111, Fax: +47 67541011
E-mail: {Einar.Dehli, Benedikte.Harstad, Thomas.Pettersen, Jan.Erik.Ressem, Are.Soerli, Geir.Waagboe}@computas.no

Abstract

SARA is an advanced Smalltalk architecture for building product model based work process assistance systems. Since 1996, SARA has been successfully used for building several mission-critical applications. The largest fielded system to date is for handling the processes of criminal proceedings in the Norwegian Police, with 8000 users. The flexibility of the Smalltalk language and environment has allowed our teams of experienced object-oriented software developers and knowledge engineers to prototype and introduce new solution concepts and AI mechanisms in the context of real world development projects. SARA supports distributed transaction handling and server-based execution in a distributed object-oriented architecture. It comes with a set of frameworks, components and patterns, and is supported by tools, methods and training programs. AI mechanisms include an agent framework, a rule inference engine and a workflow/work process execution engine.

Introduction

Many early attempts at introducing knowledge-based technology to support business processes have had little real impact on the way business is conducted. A reason for this may be that expert systems developers have had a bias towards the more challenging and complex decision problems. Though interesting in its own right, replicating the competence of experts addresses a small fraction of the work carried out in most organizations. Much higher overall payoff can be expected from successful application of knowledge technology to the plethora of routine work processes.

Since its foundation in 1985 (as Computas Expert Systems AS, or CX), Computas has been guided by a motivation to help individuals and organizations apply knowledge more efficiently and wisely. More specifically, the goals have been: (1) to free the highly skilled from the need to do repetitive work and routine application of knowledge, allowing them to spend more time on the creative and fun aspects of work; and (2) to allow the less skilled to carry out more work, at higher quality, and with higher satisfaction, thus raising their status and self-respect.

To achieve these goals, Computas has primarily recruited employees skilled in Artificial Intelligence, Object Technology and User Interaction. Many years of exposure to diverse domains and project types for these people have now yielded a family of reuse architectures for building what we may denote as Product Model Based Work Process Assistance, or EXPERT WORKFLOW™ systems. These architectures are based on a set of frameworks, components and patterns, supported by tools, project methodologies, and training programs. All architectures share common representation formats and design patterns, but are optimized for different application types and implementation environments. The focus of this paper is SARA, a flexible Smalltalk architecture, which has proven an ideal platform both for experimental prototyping of new solution concepts and AI mechanisms, and as a solid foundation for deployment of mission-critical applications. As frameworks and interfaces stabilize, they are also made available in BRIX™, an industry-standard, component-based COM architecture, and in an emerging JavaBeans/ CORBA implementation.

The EXPERT WORKFLOW™ Solution Framework

In an EXPERT WORKFLOW™ application, the user is guided to system functionality through explicit representations of business processes and work tasks. This makes the computer system a very good tool for controlling and managing the flow of work, ensuring correct execution of processes, and increasing work efficiency by providing just-in-time access to relevant information and computer functionality. The power and efficiency of the solution concept come from the combination of—and tight integration between—*expert systems technology*, *object technology* and the business process focus that *workflow systems* have.

To convey an understanding of the EXPERT WORKFLOW™ solution concept, we will briefly describe two example applications: HELENE, a dues collection system, and BL, a police support system for handling criminal proceedings.

Among the other systems implemented with the EXPERT WORKFLOW™ frameworks, we find a payroll and human resources administration system, a project and contracting administration system, and a real time system for operator support in a steam plant.

Helene

The Norwegian National Insurance Institution has established an office that handles collecting of child support from divorced parents that do not have custody, but keep economic obligations towards their children. Currently, about 65 officials are actively working to collect dues from child support debtors in Norway. The officers, who act on behalf of the children as creditors, communicate with the debtor (and the creditor) and use legal enforcement, if necessary, by taking action such as different kinds of debt collection by coercion.

Their work is governed by a very complex set of laws and regulations set up to ensure the rights of the children and the parent taking daily care of the children, as well as the rights of a debtor in case of economic difficulties on his or her part.

In the HELENE system, different tasks and procedures in the collecting proceedings are represented declaratively as procedures in the computerized workflow system. Currently there are about 100 different business process definitions in the system. Lawyers and legal experts are responsible for creating and maintaining the procedures and rules in the system, and changes of these instantly implies change in business practice.

Relevant legal rules and regulations are also represented as rules in a knowledge base. This results in effective computer support for the officials in their work, and ensures that the proceedings always comply with legally correct procedures.

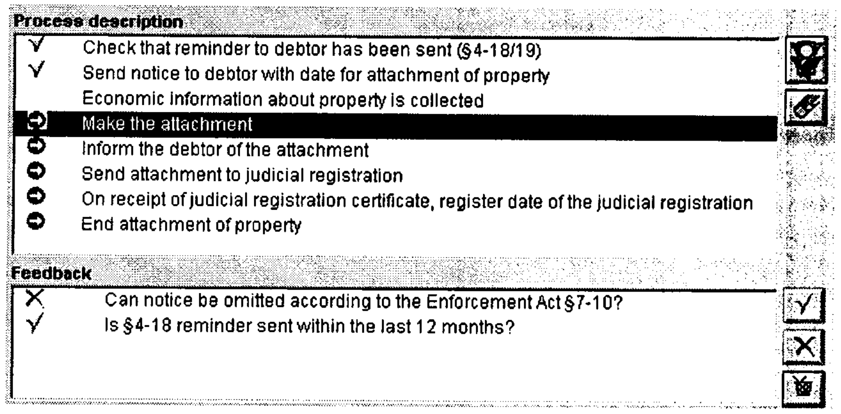
Figure 1 shows an application view during procedure execution: A parent has not paid his debt after repeated reminders, and the executive officer has started a collection procedure to attach some property of the debtor.

The top pane in the view shows the process and the activities/steps in the process. The activities marked ✓ are already executed, and the activities marked ➲ are mandatory. To execute the selected activity, the user clicks the traffic light. The light is green if all preconditions for the activity are satisfied, or else, it turns red and the activity cannot be executed.

As an example of the connection between processes and the *rule base*, we can have a brief look at the activity *"Make the attachment"*. The activity has a precondition named *"The attachment can be made"*. The precondition is tested when the officer tries to execute the activity, and this triggers reasoning in the rule system.

The lower pane is used for *feedback* and *questions* to the user. *Feedback* has two origins. It can come from the actions in the processes when something must be said about what an action did. It can also come from the rule system as information of the reasoning that is taking place. *Questions* occur when the rules don't get sufficient facts from the domain model to evaluate. In this case, the user must provide these facts manually. The questions can be answered with yes (✓) or no (✗). In the example in figure 1, the question *"Is §4-18 reminder sent within the last 12 months?"* comes from some rule reasoning originated in some condition of the first activity, *"Check that reminder to debtor has been sent"*. The question *"Can notice be omitted according to the Enforcement Act §7-10?"* has its origin in some condition of the second activity.

The actions in the procedure steps (or business process activities) in HELENE, among other things consist of creating letters to the involved persons and authorities, such as a notice to the debtor and judicial registration of the attachment to the proper authorities. An action that causes creation of a letter opens the associated word processor and fills out document fields with data from the domain model in the relevant template document.


Process description

✓ Check that reminder to debtor has been sent (§4-18/19)
✓ Send notice to debtor with date for attachment of property
Economic information about property is collected
➲ Make the attachment
➲ Inform the debtor of the attachment
➲ Send attachment to judicial registration
➲ On receipt of judicial registration certificate, register date of the judicial registration
➲ End attachment of property

Feedback

✗ Can notice be omitted according to the Enforcement Act §7-10?
✓ Is §4-18 reminder sent within the last 12 months?

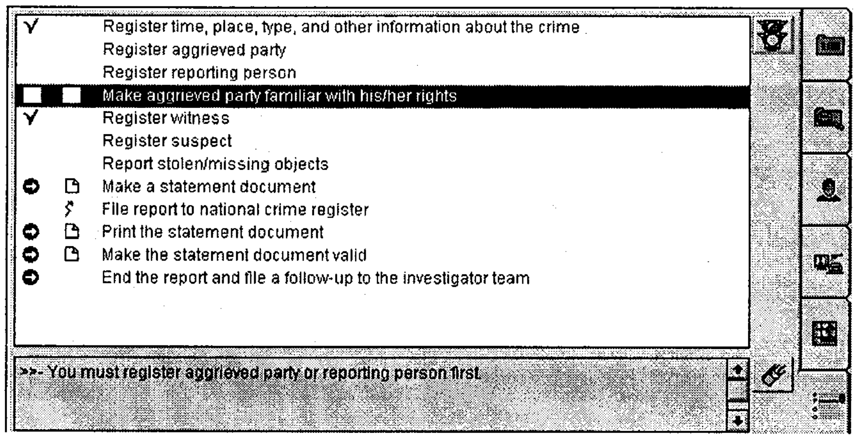
Figure 1 Application view from HELENE, illustrating how SARA assists in the execution of a debts collection procedure

HELENE is implemented in Smalltalk as a client application and uses a relational database system on a server for data storage. It also communicates with a back-end accounting mainframe system. It interacts with a standard word processor for handling documents, and with a hypertext tool for detailed explanation of each step in the processes. These explanations contain hypertext links to relevant legal rules and regulations.

HELENE was the first system based on the EXPERT WORKFLOW™ concepts. The system was put into operation in May 1995 after a 9-month development period, including specification, knowledge acquisition, implementation, testing and education. The users had limited experience with modern computer systems, and very few of them had experience with collecting debt procedures. After about a year in operation, the officials supported by HELENE have reduced the average proceedings time from about 3 months to less than 3 weeks, and work quality has increased significantly.

BL

BL is an EXPERT WORKFLOW™ system used by the Norwegian Police Department and the Public Prosecuting Authority for handling the processes of criminal proceedings. Compared to the HELENE system, the domain model of BL is more complex. In brief, it covers criminal cases with different kinds of related information, including involved persons and their roles in the cases, objects that are missing or stolen or act as evidence, etc.



```
✓   Register time, place, type, and other information about the crime
    Register aggrieved party
    Register reporting person
    Make aggrieved party familiar with his/her rights
✓   Register witness
    Register suspect
    Report stolen/missing objects
    Make a statement document
    File report to national crime register
    Print the statement document
    Make the statement document valid
    End the report and file a follow-up to the investigator team

>>- You must register aggrieved party or reporting person first.
```

Figure 2 Work process in BL, showing the steps to perform when someone wants to report a crime at the police station

Figure 2 shows one of the about 70 work processes in the initial BL release, the process performed when someone wants to report a crime at the police station.

There are some differences between how processes behave in HELENE and in BL. In contrast to HELENE, the activities in a process in BL do not normally have to be performed in a strict sequence of order. In addition, most activities can be performed several times; e.g. when there is more than one witness in the example above. In HELENE, each activity can only be performed once. The workflow component in EXPERT WORKFLOW™ takes general business differences like these into account.

Putting the system into operation in the police and the Public Prosecuting Authority all over Norway is a substantial organizational task. After a 9-month development period, this task was started in April 1996. At the end of 1997, the system was in full operation in 30 police districts, together covering 3500 end users. The remaining 24 districts will be put into operation incrementally until the end of 1998. About 8000 people, including policemen, prosecution attorneys and office workers, will then use the BL application.

SARA Architecture Overview

The first SARA-based EXPERT WORKFLOW™ systems consisted of three interrelated models, each representing different aspects of business knowledge:

- A representation of *business processes* and a workflow engine that manages the execution of these processes. We call this representation the *process model.*
- A representation of the *substance* in the business domain in a traditional object model with services/functionality operating on this model. We call this representation the *domain model.*
- A representation of the *business rules* as a *rule base* with an inference engine.

In figure 3, the relationships between these three knowledge representation models are illustrated. The three representations work together in order to provide knowledge-based computer support for business processes. Thus, the starting point is the *process model.* The representation of an *activity* in a business process sends *messages* to the *domain model* to provide such support. These messages are called *actions.* The *rules* act as execution conditions for the processes and activities. A *precondition* must evaluate to true if a process or an activity is allowed to start. An *action choice condition* is used to determine which set of actions is performed in an activity. A *postcondition* is evaluated to ensure that the process is in a legal state before it is marked as correctly executed.

To evaluate, the rules need *facts.* They get the facts from the domain model. They can also ask the processes for facts. To the rules, the process model in this sense can be regarded as part of the domain model. Rule reasoning can produce *side effects* if the rules are set up to send messages to the domain model and process model when evaluated.

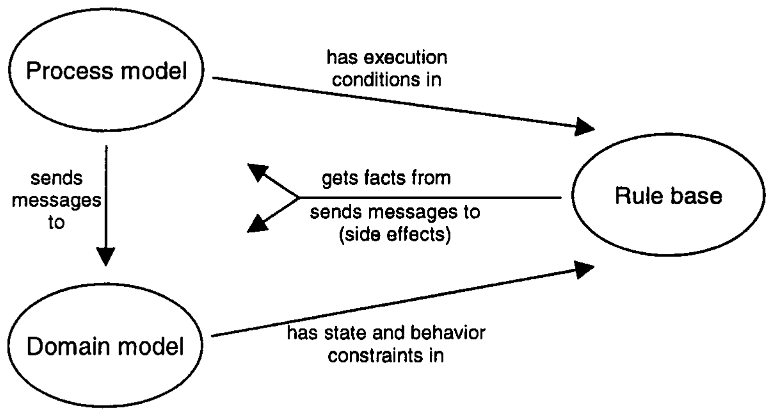

Process model
has execution conditions in
sends messages to
gets facts from
sends messages to (side effects)
Rule base
Domain model
has state and behavior constraints in

Figure 3 Relationships between the three knowledge representation models found in early SARA systems

The rules are not only conditions for the processes and activities, but also act as *constraints* on the *state and behavior* of the domain model.

Figure 4 shows the separation between the SARA reuse frameworks and the application specific parts of an EXPERT WORKFLOW™ application. With SARA, most of the application logic can be specified declaratively. The elements that must be built are the domain model, the business processes, the rule base and the application specific views. In a real world situation, the picture is somewhat more complex. There will always be situations where modifications to some elements in the reuse frameworks are required for proper behavior. In an open environment like SARA, this can easily be done through classical object-oriented inheritance at key points in the frameworks. It is also straightforward to integrate other components into the application, e.g. for communication with a mainframe system. The aforementioned HELENE and BL systems both communicate with mainframe systems, through different communication components.

As the SARA frameworks have matured, the process of building EXPERT WORKFLOW™ applications increasingly has shifted its focus from a traditional object-oriented development effort towards more of a *knowledge acquisition* and *representation* task. (We hold the view that the object-oriented analysis/design involved in building a domain model is just one aspect of acquiring and representing knowledge). With SARA, it is possible to do this process in an incremental and very rapid manner. As soon as the first few processes and a part of the domain model are defined, they are instantly implemented in the system and act as a basis for discussions and further development. Editing the processes and rules is done with *authoring tools*, as illustrated in figure 4. Object-oriented CASE tools can be used as authoring tools for the domain model.

EXPERT WORKFLOW™ is conceived from a knowledge engineering tradition. An essential strength of SARA compared to other object-oriented application frameworks, is that the application specific business processes and rules are declaratively represented as *data* (i.e. object instances) and *not code*. This means that business processes and rules are defined and maintained separately from application code. For the systems developed with SARA, this quality has been one of the most valuable, especially over time when the system is in operation and continuous maintenance. To some extent, the definition of the domain model is also represented as data in a meta model. However, objects with application specific *behavior* require classes and methods, i.e. code, to be added. In fact, even the definitions of some of the simpler user interface views are separated out as data, and the contents and layout of these views are built dynamically from that data at runtime.

SARA is an evolving architecture. The latest release of SARA is based on a 3-tier client-server architecture, as illustrated in figure 5. The three tiers are:

- The client tier is the part running on the end user's computer. The client tier can be a Smalltalk application, a Java application, a Java applet running in a Web browser, or pure HTML-pages accessible from any Web browser. In one system, there could be one or more of these client types.
- The application server tier is built with a high performance, object-oriented, transactional/ multi-user application server and database.
- The enterprise and data storage tier consists of the object storage facilities, a relational database for backend data storage (can be omitted), and other systems and data sources in the enterprise.

Even if this architecture is called 3-tier, it fits well into an n-tier architecture when taking into account a total enterprise architecture consisting of many different systems and applications.

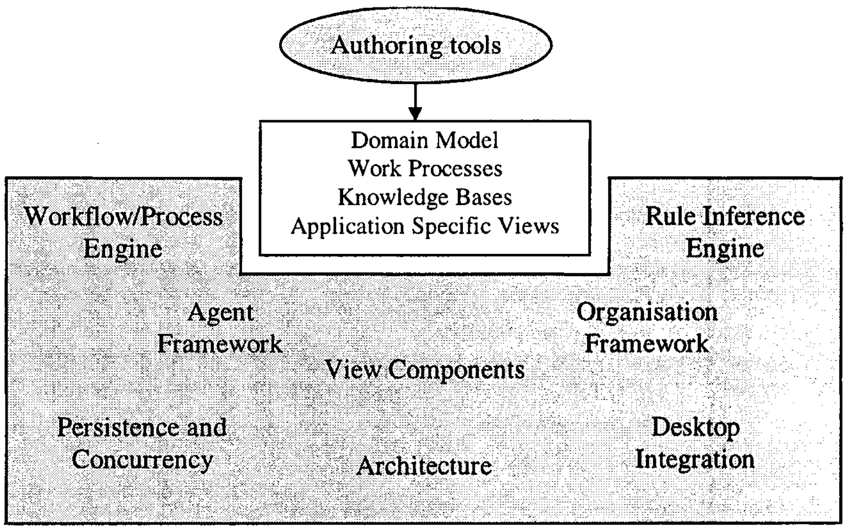


Figure 4 *Illustration of how authoring tools are used to build application specific knowledge bases in the context of the SARA architecture and reuse frameworks*

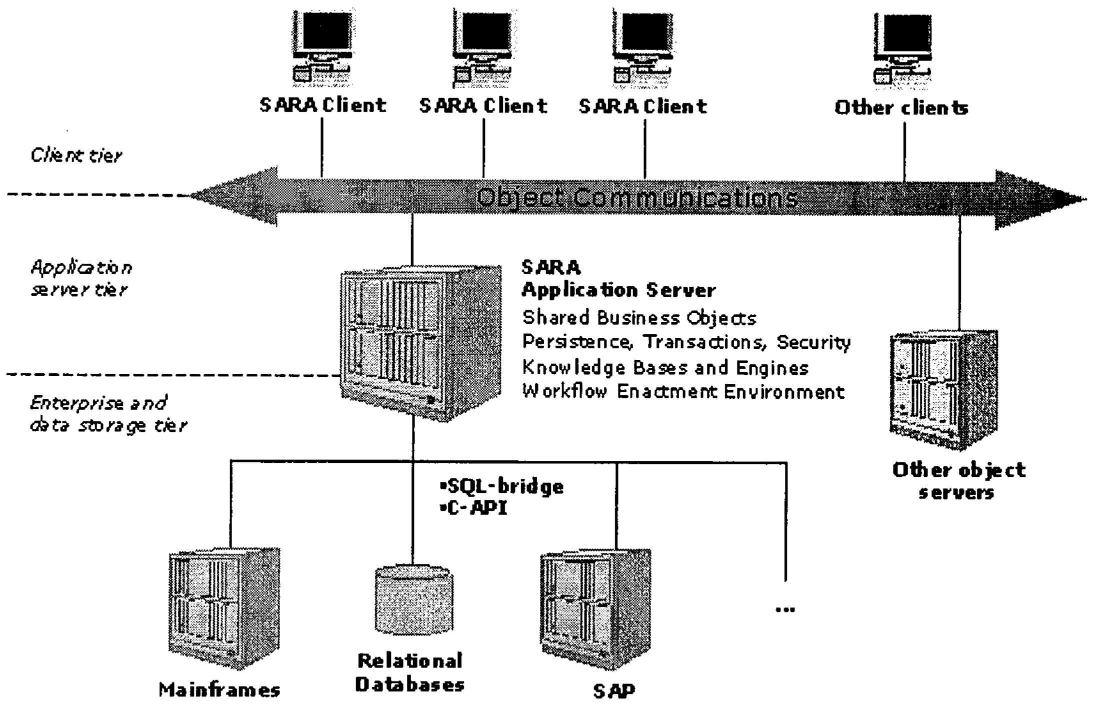


Figure 5 SARA multi-tier architecture

AI Mechanisms in SARA

In addition to the knowledge representation and reasoning mechanisms described above, we have incorporated forward-chaining rule inference and an agent framework into the latest release of SARA. Further plans include case-based reasoning and constraint satisfaction techniques.

Rule Inference Framework

The rule inference framework contains the means to store a rule base, as well as mechanisms for evaluating logical sentences using the rules (the rule engine). A typical rule is of the form:

if premise **then** consequent

Both the premise and the consequent are *predicates*. The rule expresses that if the premise is true, then so is the consequent. A more concrete example is:

```
if   |( ?Case.hasInvolved(?Person),
        ?Case.hasInvolvedAmong(?Person.relatives) )
then  §DisqualifiedAsWitness(?Person, ?Case)
```

In this example, ?Case.hasInvolved(?Person) is an *atomic predicate,* whereas §DisqualifiedAsWitness (?Person, ?Case) is a *consequent predicate.*

Predicates can be thought of as syntactic entities expressing something about the world; as Owns-a-Car, Is-Red, Has-Gone-Fishing, Is-Married-To, etc. If predicates are to be useful, we need variables as well: i.e. Owns-a-Car(?Person), Is-Married-To(?Person1,?Person2). Question marks indicate variables. Given a certain binding of the variables to objects in the domain model, a predicate is always true or false.

Our rule language is more or less the language of ordinary first order predicate logic with the quantifiers removed. This can also be called the language of quantifier-free predicate logic. In predicate logic the basic building blocks in the language are predicate letters and terms. Atomic formulas are built up from predicate letters and terms, and these again can be combined with the connectives to build more complex formulas. In our object representation of rules, both the formulas and the predicates are represented by the same objects, and we have chosen to call these objects *predicates*. Thus we have ended up using the term *predicate* as synonymous with the term *formula.* The set of terms of our rule language consists of variables and *object paths*, i.e. declarative object expressions that can be stored persistently, and are dynamically bound and invoked at runtime, providing a means of referring to or altering the state of domain objects.

Agent Framework

The agent framework can be seen as architectural glue for a distributed, knowledge-based system. It serves two main purposes:

1) It is a *skeleton framework ("white box")* for constructing system components that handle knowledge. An

agent in SARA can be seen as an object that is responsible for obtaining, acting on, and distributing knowledge about other objects in the system. Agents are connected to an environment (e.g. some domain objects), typically have a knowledge base, and communicate and collaborate with other agents via a well-defined agent communication language. In this way, the agent framework allows new knowledge representation and reasoning mechanisms to be added to an application in a principled manner.

2) It serves as the *principal layer ("black box")* for handling distribution of, and communication between, existing knowledge representation and reasoning mechanisms in the SARA framework. Distribution of knowledge bases is crucial for handling scale, complexity and distributed control of continuously evolving knowledge in an enterprise-wide system. When we want to distribute rules over many different rule bases, or when a process needs to evaluate a rule, or when a rule is dependent upon some object in the business model, it will be the task of some agent to handle knowledge interchange between these different knowledge components.

The agent framework in SARA consists of:

- *Basic agent substrate,* which can be extended ("white box").
- *Agent facilitators,* for co-ordinating agent communication.
- Currently three types of *reusable agents* ("black box"): *rule agents*, which manage rule bases, *process agents*, which manage work processes, and *business model censoring agents*, which monitor changes in the business model.
- An implementation of a *standard agent communication language*, KQML, for querying and manipulating agent knowledge.

Conclusions

Our experiences with building intelligent workflow systems show that:

- A dynamic, flexible and mature object-oriented language like Smalltalk is very well suited as a tool for building knowledge-based systems.
- Knowledge-based systems technology can benefit significantly from being tightly integrated in a general-purpose object-oriented programming language.
- Combination of these two technologies can be used efficiently to develop intelligent workflow systems that apply well to a large range of real-world problems.

Smalltalk has proved to be very well suited for implementing the desired mechanisms, including the workflow and rule inference engines. The level of abstraction and the dynamic capabilities of the language are two important reasons for making this efficient and relatively straightforward. Other dynamic high level object-oriented languages, such as CLOS, Self or Dylan, could also have been used, but they lack Smalltalk's level of maturity in client/server commercial tools, professional development environments, extensive libraries for integrating with other systems, etc.

Authors

Einar Dehli is co-founder of Computas AS in 1985, and is currently Vice President and Process Owner, Technology. He has a M.Sc. degree in Computer Science from the Norwegian University of Science and Technology (NTNU) in Trondheim. In addition to being a seasoned Lisp and Smalltalk developer, with experience from both research and commercial projects, he has held various company management positions He recently served as Project Manager for the latest SARA release.

Benedikte Harstad Kallåk has a M.Sc. degree from the University of Colorado. She has worked with expert system technology for 5 years, and has 4 years experience with SARA development. She has also been responsible for incorporating the techniques from EXPERT WORKFLOW™ into Microsoft's COM™-technology.

Jan Erik Ressem has a M.Sc. degree from the University of Oslo. He has studied and worked with object-oriented methodologies and databases for 4 years, and has 3 years of experience with SARA development and expert systems. He is currently responsible for the multi-tier and distributed aspects of the SARA architecture.

Thomas Bech Pettersen has a M.Sc. degree from NTNU in Trondheim. He has more than 12 years of experience with knowledge engineering. He has worked with implementation tools like Gensym's G2™, Neuron Data's Smart Elements™, CLOS, Smalltalk, KEE™, etc, and has participated in European research projects working on methodologies for knowledge acquisition. He is one of the most influential architects behind the EXPERT WORKFLOW™ solution concept.

Are Sørli has a M.Sc. degree from the University of Oslo. He has studied and worked with AI and object-oriented methodologies for 4 years, and has 2 years of experience with SARA development. He is the primary architect behind the agent framework in SARA.

Geir Waagbø has a Ph.D. degree in Mathematical Logic from the University of Oslo, and a M.Sc. degree in Computer Science from NTNU in Trondheim. He is the primary architect behind the new rule inference engine in Sara.