Selecting components in large COTS repositories

Justin Clark a, Chris Clarke a, Stefano De Panfilis b, Giampiero Granatella c, Paolo Predonzani a, Alberto Sillitti c*, Giancarlo Succi d, Tullio Vernazza c

a BTexact Technologies, Orion Building, Astraal Park, IP5 3RE Ipswich, UK
b Engineering Ingegneria Informatica S.p.a., Via San Martino della Battaglia 56, I-00185 Roma, Italy
c DIST—Università di Genova, Viale Causa 13, I-16145 Genova, Italy
d Faculty of Computer Science, Center for Applied Software Engineering, Free University of Bozen, Dominikanerplatz 3 Piazza Domenicani, I-39100 Bolzano, Italy

Received 3 April 2002; received in revised form 8 February 2003; accepted 10 September 2003
Available online 30 December 2003

Abstract

The growing availability of COTS (commercial-off-the-shelf) components in the software market has concretized the possibility of building whole systems based on components. In this multitude, a recurrent problem is the location and selection of the components that best fit the requirements. Commercial repositories that offer search mechanisms have reduced these difficulties: system integrators can rely on a wider variety of components and can focus better on the composition of systems. The size of the repository can be an initial obstacle but iterative approaches allow integrators to familiarize with the repository’s structure and to formulate effective queries. This paper discusses the search techniques in CLARiFi, a component broker project that supports integrators in the selection of components for systems.

Keywords: COTS repositories; Component selection

1. Introduction

In recent years several commercial repositories with public access have started to publish information on available components and to work as a commercial channel for their distribution. These COTS market-places are meeting points between component suppliers and system integrators. They have deployed and set off the concept of reuse at a commercial level, thus contributing to the diffusion of component-based software engineering (CBSE) practices.

Existing COTS repositories currently contain a few hundred components. Having been developed as online intermediaries or brokers, the repositories have grown as business-to-business (B2B) systems on the Internet. In the B2B market, the size expectations for 2001 in Europe were below 200 M-Euros, but the predictions for 2004 are above 1200 M-Euros (Durlacher, 1999). OECD reports similar growth ratios for North America (OECD, 1999). The interest in COTS and the development of technologies and infrastructures (Szymerski, 2002; Krieger and Adler, 1998)—like Enterprise Java Beans and the .NET platform—set high expectations on the future availability of numerous components on public repositories.

In addition to public repositories, there is a different, complex reality of component production and reuse within single companies. IBM reports that in 2000 the San Francisco framework contained more than 2000 components (Van Emde Boas-Lubsen, 2000). Also, many large companies have huge sets of reusable software assets, which they frequently organize in their knowledge-management systems. These assets may not be strictly components—for instance, they can be legacy

0164-1212/$ - see front matter © 2003 Elsevier Inc. All rights reserved.
subsystems that are not developed using component-based technologies—but they are frequently used as “building blocks” in the development of complex systems (Van Der Linden and Müller, 1994).

Users of repositories face the problem of searching in wide sets. Repositories address these problems by providing automated selection mechanisms. Existing mechanisms include categorization, search by aspect, and full-text search (Henninger, 1997). Whatever the mechanism is, users need to understand and master the repository’s structure in order to formulate effective search queries.

When developing repositories and search tools, the integrator’s perspective—namely, building systems—needs to be kept in mind: a component-based system is built from existing components that work together through adaptation and integration. The difficulties of single component selection multiply in the case of the selection of many components for a system. This “system” perspective will be adopted throughout this paper.

In general terms, the repository’s task is to find a set of components that cover the functional, non-functional, technological, environmental, and compatibility requirements of the desired system.

The concept of system may vary. Several COTS repositories consider as system the component platform (COM, EJB, etc.). Consequently, the system is a technological framework that exposes empty slots, which are filled in by the components. In another vision, less bound to specific platforms, the system is primarily a collection of functionality, which the integrator articulates at a conceptual level and the repository fills with actual components.

The work here presented has been developed in the EU-funded project CLARiFi (CLear And Reliable Information For Integration), which involves eight European partners. The paper is organized as follows: Section 2 presents the perspective and the fundamental hypothesis adopted for an effective selection of components; Section 3 analyzes the specific issues and solutions taken in CLARiFi; Section 4 and subsections present a case study comparing the CLARiFi system with the market leader, Component Source; Section 5 discusses the lessons learnt; finally, Section 6 draws the conclusions.

2. Component search: the “funnel” metaphor

The interest of integrators in a COTS repository greatly depends on its ability to offer a manageable and trustworthy view of its content. The increasing availability of components makes the search in large repository a necessity of the near future, if not of the present. Also the monetary and intellectual investment that is associated with the adoption of components justifies the accuracy that integrators expect from searches in repositories.

What a manageable view means depends on the characteristics of the repository. For small repositories, browsing the whole content to look-up a component can be a manageable view. For larger repositories, a more complex and automated interface may be necessary and may justify the extra effort in understanding the mechanics of the search.

When using a repository, the best search technique depends on the structure of the repository itself (Prieto-Diaz and Freeman, 1987). Several authors in the literature advocate that integrators need to master the structure and mechanisms of the repository before being able to effectively select components from it (Henninger, 1994, 1997; Frakes and Pole, 1994; Damiani et al., 1999). More precisely, the integrator needs to understand its structure in the context of his objective (building a system), finding a match between the conceptual level—the system’s architecture—and the concrete level—the presence of components in the repository. In general, CBSE is a “best-fit” problem, where the integrator finds an acceptable compromise between the requirements, the system architecture and the design of the available components.

It is acceptable that, to find the best compromise, the integrator goes through several iterations, understanding progressively the repository and formulating better queries. We call this iterative process the “funnel” metaphor of selection. The large side of the funnel corresponds to the generic queries that return a much too high number of candidates; the small side of the funnel corresponds to the too narrow queries that return an insufficient number of components to cover the requirements. It is not only a matter of number of components: it is also a matter of quality and fitness for use. Ideal searches are those—positioned somewhere in the middle of the funnel—that return manageable sets of components but large enough to embrace the system view of the integrator.

As the integrator refines the queries, he clarifies his own view of the desired system and of the repository. The clarification leads to more focused queries. In component-based software engineering it is acceptable that selected components be adapted to the design (Ostertag et al., 1992); it should also be acceptable that the design—i.e., the integrator’s view of the system—be adapted to the actual availability of components.

Adjustments of the queries may be necessary for several reasons:

- The number of candidates can be wrong: either unmanageable or too narrow.
- The components may be not adequate due to some discrepancies between their description and what they actually do.
• The components may not fit together due to incompatibility.
• The system design may not be the right view to decompose the problem in parts and to look up components that fit the parts.

With the iterative approach the integrator works up and down the “funnel” until the results match the requirements. The integrator has to work more in the specification of the query, articulating better and more explicitly what the system requires. Once the right query is formulated, it makes up a—more or less detailed—requirement analysis for the system. The effort is justifiable as long as the complexity of the query is commensurate to the complexity of the system. This is usually the case because the integrator is keener to refine and better specify those parts that are conceptually complex and more difficult to match against the real components.

From the repository’s perspective, the final query is tailored on the characteristics of the repository and specifies in detail those aspects that better discriminate between the candidates. From the integrator’s perspective, the final query is a specification of the desired system and articulates those aspects that are necessary to match the abstract requirements to the reality of the components.

3. The selection process in CLARiFi

Several existing repositories implement advanced search techniques. However, the target of their searches is a single component. These repositories help integrators find the individual pieces but do not consider how the pieces contribute to build complex systems. For integrators, this approach is insufficient.

The CLARiFi project was started with the purpose of selecting components for systems, thus better embracing the integrator’s perspective. The consortium behind CLARiFi includes eight partners from the industry, the certification bodies, the research institutions and the academy. The partners are Engineering Ingegneria Informatica, British Telecom, TUEV Nord, Delta, ENEA, Keele University, University of Durham, Università di Genova. CLARiFi is a component broker implementing a supplier interface for the registration of components and an integrator interface for the selection of components. CLARiFi allows integrators to:

• search components,
• focalize on the system during the articulation of the requirements,
• progressively understand the repository’s structure and the characteristics of the available components.

The details of CLARiFi selection process are explained in the following sections.

3.1. Functional and non-functional requirements

CLARiFi supports the integrator in the funnel metaphor. In this process, the functional requirements typically make up the first rough-out query. Functional requirements describe what operations the system must do. They are expressed in terms of a variant of the \(\text{action, object, medium} \) tuples used by Prieto-Diaz and Freeman (1987). Such “functional abilities” are additive and decomposable at several levels of detail. The result of the first query is the set of components that fulfill at least one functional ability.

Ideally, the functional requirement could be the one at the coarsest level of detail. However, it is unlikely that there exists a single component that fulfills it. Moreover, if there were more than one candidate, the single requirement would be insufficient to discriminate between them. If a certain level of detail is not adequate, the specification of the functional requirements can break down at a finer level of detail. The process can take place recursively, breaking down the functional requirements for which components are insufficient or inadequate (Fig. 1).

The functional requirements move the selection from the narrow to the large part of the funnel. The integrator specifies functional requirements to broaden the candidate set in several steps:

• Definition of the high-level functionality. It is very likely that the broker will find no candidates.
• Decomposition of the high-level functionality into more detailed ones (broad-range decomposition). Some functionality will probably have matching components.
• Decomposition of the unsatisfied functional abilities (targeted decomposition). In a few iterations, possibly all functionality can be covered.

Fig. 1. Example of functional decomposition.
After the first rough-out query, non-functional requirements come into play to move the selection towards the narrow part of the funnel. Non-functional requirements can regard performance, licensing and cost, reliability, resource usage, etc. Non-functional requirements may apply to the system as a whole or only to parts according to the functional decomposition. The management of non-functional requirements depends on the requirement’s type: some of them are propagated to sub-functionalties (e.g. constraints on operating system or certification), others are split in order to satisfy the constraint on their composition (e.g. constrains on the price of the final system or system response time).

The example in Fig. 1 shows a branch of functional decomposition of an e-commerce management system. The integrator can specify non-functional requirements for every node of the tree. The CLARiFi system manages these requirements in different ways according to the specific type of requirement. Fig. 2 shows three non-functional properties applied at different levels of the decomposition tree. The first inserts a cost constraint at the root level, this means that the sum of the costs of single components must be below $1000. The second inserts a response time constraint only in the Order Management subsystem, this means that only components that belong the specified subsystem have to fulfill the requirement, and the sum of the execution times of the single components must be below 5000 ms. Finally, the third inserts a certification constraint on the Payment Management subsystem represented by a small ribbon on the left of the specified node. Not all non-functional requirements specified by the integrator are showed in Fig. 2. As instance, no constraints either on the operating system or on the components’ technology are showed, even if the CLARiFi system considers them during the ranking and selection process.

3.2. User profiles

The selection process yields systems that are the most suitable given the integrator’s specific characteristics and goals. Different integrators approaching the same problem would probably select systems made of different components. The idea of “user profiles” is that there are certain recurrent requirements that characterize and distinguish each integrator. More generally, also categories of integrators—for instance, those that work in the same domain—may have recurrent requirements. User profiles explicitly describe such requirements and make them reusable in several selections. User profiles have several rationales:

- User profiles are similar to protection profiles, which are functional and non-functional requirements encapsulated in a standard definition. Standardization bodies typically define protection profiles, but also companies and groups of companies can have their own protection profiles.
- Integrators may need certified components. For instance, this happens in certain safe-critical applications. The use of certified components—which may be limited only to some functionality—is a recurrent constraint for such integrators.
- Integrators may have non-functional requirements. These typically regard performance and need to be met in real-time applications or where contractual Service Level Agreements (SLAs) define the system performance.
- Integrators may have technological constraints on the type of components: e.g. EJB vs. NET.
- Integrators may be bound to select components only from a list of preferred suppliers.

User profiles are containers of standard or recurrent requirements that exist beyond single selections and that can be reused—more precisely, imported—on the user’s demand.

3.3. Ranking of results

The definition of requirements breaks down the space of the problem at the right level of detail and imposes constraints to limit the candidates to those that have certain mandatory, fundamental characteristics. The definition of requirements eventually yields an almost stable set of candidates, among which several alternatives may exist.

A ranking process determines which components, among the alternatives, are the best according to some secondary characteristics. Integrators define such characteristics as preferences rather than as constraints. The specification of preferences uses the same schema as the constraints (based on functional and non-functional characteristics). The ranking process uses multiple-criteria decision-making (MCDM) techniques to deal with the several, possibly contradictory preferences and to evaluate the trade-offs between them.

The simplest case of ranking is by a single property such as price, a more complex one include different kind
of properties that often are hard to compare because they use different measure units (e.g. memory required and price). Moreover, some properties can be qualitative. In such cases, a conversion into scalar properties is required because only scalars can be compared. If qualitative properties are associated with an ordinal scale, it is possible to assign a range in such scale (e.g. Low associated with [0, 3]; Medium with [3, 6]; High with [6, 10]). If the property has a nominal scale, the only way to compare them is through a Boolean distance (0 or 1) (e.g. the operating system or technology attributes: if the integrator chooses Linux, the associated values will be 1, and 0 will be associated to other operating systems).

There are two sets of MCDM criteria to approach this problem: the former does not require any human interaction, but less accurate; the latter require some human input, but are more accurate. The CLARiFi system implements the following MCDM criteria without human interaction: maximin and maximax. The former order components according to their worst property value (pessimistic approach); on the contrary, the latter order components according to their best property value (optimistic approach). The MCDM with human interaction implemented are: saw and linmap. The former requires, from the integrator, a set of weights, one for each property, used to compare components; the latter, requires the same set of weights of the saw method and property values of an ideal component used in the comparison minimizing the distance from the ideal one.

The CLARiFi system provides different way to insert both constraints and preferences including text and graphical based (Fig. 3).

Fig. 3 shows a real example of graphical interface used to collect input from integrators. Each vertical line represents a property and markers delimit the acceptable range of values for the specified property. Components are represented as sets of segments that identify a path across the properties. Intersections between a component line and the properties lines identify the component specific features. Components that satisfy integrator’s requirements are painted with black lines, the others with light gray.

Of course, all the ranking methods can fail in extreme conditions, but they works quite well in common situations helping the integrator in the choice. Moreover, ranking is not binding for the integrator: a highly ranked component is not always the one that best fits the actual integration. The value of ranking is in the short-listing of the candidates, on which the integrator can later make further investigations, for instance, through trial of demo versions.

### 3.4. Compatibility

Compatibility ensures that components short-listed as good candidates can actually cooperate to build a system. Compatibility is a further criterion on selection. Integrators may rank higher a set of compatible components. However, integrators may also overcome incompatibility through adaptation, wrapping, and gluing, although this usually implies a higher effort during the integration.

CLARiFi checks compatibility by looking at the communication paths between components. Each component is a black box, which exposes its functionality through ports (Van Der Linden and M€uller, 1994; Borgida and Devanbu, 1999). Conceptually, a port is a communication stack—much like the OSI stack—that schematizes the protocols, APIs and formats through which communication takes place (Fig. 4).

The layers of a port may implement the link and be closer to the physical aspects—function and method invocations, parameter passing, networking—or may contribute to define the semantics of the messages that are transmitted. For compatibility to be possible, the component ports need to be similar in structure and expose the same link and semantics layers. The compatibility test verifies that compatible ports exist between components and reports the results to the integrator.

Usually, integrator is interested only in compatibility between ports of components that have to communicate,
for this reason the CLARiFi system allows an integrator to choose ports and components to be checked for compatibility.

The compatibility check requires some extra effort to components' suppliers during the classification process but it provides a powerful mechanism to retrieve only useful components from a huge set.

3.5. Multiple selections

Multiple selections allow the integrator to branch the search into several alternative selections that lead to different solutions.

The integrator may start from different hypotheses in the requirements and see to which different solutions each hypothesis leads. Also frequently, at various stages of the selection, integrators pin down some primary decisions and check how the rest of the system “rotates” as a consequence of different secondary decisions.

Multiple selections take into consideration different selection approaches and evaluate them at the end, under a global perspective. Given some alternative components, the integrator can create a selection branch for each of them, thus opening the possibility of different final systems. The choice of the actual component to adopt is deferred until the other components are chosen and the alternative components can be evaluated in the context of the overall selection.

Fig. 5 shows different functional decompositions of three functionalities offered by a system. Functionality F0 could be considered either a single functionality or it can be decomposed into two more specific functionalities; functionality F1 can be decomposed in different ways, either into two or three sub-functionalities; functionality F2 is not decomposed, so no alternatives are possible. According to this decomposition, the CLARiFi system searches for components to satisfy the different functional decomposition and, finally, proposes components for four candidate systems: (F0, F1, F2), (F0, F1', F2), (F0', F1, F2), (F0', F1', F2).

At the end of the process, the integrator can choose the system among the ones proposed.

4. A case study

To validate the proposed tool and methodology, an objective evaluation is required. Such evaluation should include data regarding the tool usage, the quality of the taxonomy used, and the comparison with the market leader.

4.1. The tool evaluation

The first evaluation is about the CLARiFi tools and how users feel about them. There were three evaluation based on prototypes at different levels of development. All of them were performed using components retrieved from Component Source. There are two main user interfaces in CLARiFi: supplier interface and integrator interface. The first one was evaluated providing to IT experts some text describing components and asking them to insert these components into the CLARiFi system. The second one was evaluated asking IT experts to (a) retrieve the best component satisfying some requirements and (b) retrieve components to build an entire system. The third one was evaluated asking IT experts to certify some components already stored in the system.

The results of the supplier interface evaluation are the following: the system requires a lot of effort for both learning how to use the tool and for insert components’ data into the system. It is quite hard to manage the built-in taxonomy, as discussed in the following sub-section. Even if the system is not very user friendly, the overall judgment is positive and after a while, classifying components in this way provides a big improvement to the search of components.

The results of the integrator interface evaluation are the following: even if users have to change their traditional approach to the search of components and leaning how to use the tool, they experiment a big potential and a very rich set of information that can be used to compare and restrict the number of retrieved components. The ranking and selection mechanism allows users to improve the selection process, even if the user interface is not very user friendly and it should be improved.

4.2. The taxonomy

The second evaluation, performed by British Telecom, is about the effectiveness of the taxonomies used inside the CLARiFi system. The aim of this evaluation is the collection of data regarding how different taxonomies affect the CLARiFi usability.
The experiment involved three e-commerce taxonomies—RosettaNet, IDC, and CLARiFi’s supplier taxonomy. According to the results, RosettaNet, then, IDC, and finally, CLARiFi’s supplier taxonomy achieved the best scores. RosettaNet is the best one mostly because of its rational structure, making it predictable, easy to browse and remember, but it manages only one domain—e-commerce. The main defect of CLARiFi’s supplier taxonomy is the aggregation of various component taxonomies without much coordinated rationalism. Therefore, there are overlapping categories, inconsistent terminology, and a not well-defined tree structure.

Collected data shows that RosettaNet is significantly better than the CLARiFi’s supplier taxonomy both applied inside the CLARiFi tools. This is not a surprise, since the former is the result of a wide consortium’s agreement while the CLARiFi consortium, without participation of any domain experts, developed the latter internally. The CLARiFi’s supplier taxonomy has the largest number of categories that provide confusion rather than help. Moreover, according to collected data, CLARiFi’s supplier taxonomy often produces a false perceived confidence about results. That does not happen with RosettaNet, which produce the highest agreement between perceived confidence and actual results.

This experiment finds out the importance of high quality and domain specific taxonomies to achieve a successful components’ search. Fortunately, taxonomies are not hard-coded in CLARiFi tools, therefore, domain specific taxonomies, such as RosettaNet, will be included into the system.

4.3. The comparison

The comparison with the market leader, Component Source (www.componentsource.com), should be done considering both offered functionalities and the results collected addressing the same problem using the two systems.

The comparison between main features of the two systems is summarized in Table 1.

The latter comparison requires a lot of effort because it required a quite big set of components that should be added to the system to make the experiment relevant. Moreover, detailed data required to perform a ranking of selected components in CLARiFi are hard to find or even not available at all. Often, these data are available only requiring further information to the producer. All these difficulties make the direct comparison experiment hard to implement. Therefore, it is still an in-progress activity, even if the UE-founded project is over. Result of this experiment will be published as soon as they become available.

5. Lessons learnt

The type of classification is the first decision of a repository and entails the possible selection techniques. The classification must have a semantic richness to improve the comprehension among suppliers, the broker, and integrators. Rich classifications open the possibility for several search techniques, from the simplest for the user—like free-text on the whole component descriptions—to the most semantically aware ones—like the functional decomposition.

Property values that suppliers provide are often qualitative. There are no scale or proximity measures to compare them. Therefore, MCDM methods are useful only on those properties that have an ordinal or ratio scale. In some cases it is possible to define simple scales—e.g. “like” and “dislike” sets—for discriminating property values.

Integrators attach more importance to certified components and properties. Certification improves the

<table>
<thead>
<tr>
<th>Feature</th>
<th>Component Source</th>
<th>CLARiFi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification</td>
<td>Simple</td>
<td>Rich, it includes the Component Source classification</td>
</tr>
<tr>
<td>Multiple points of view</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>System point of view</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Component point of view</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Effort required to classification</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Information for integrators</td>
<td>Low, except when use case are provided</td>
<td>High</td>
</tr>
<tr>
<td>Comparison of attributes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Ranking of components compared to an ideal component</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Alternative selection and rollback</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Search and selection refinement</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Storing of a context</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Certification of components</td>
<td>No</td>
<td>Yes, supported at different levels</td>
</tr>
</tbody>
</table>
perceived trustworthiness in the components and in the repository. As a derived implication, integrators prefer components that they have already used, since these are proven—and in a certain sense certified—to be reusable. Repositories should show to integrators who—the supplier on a certification body—provides and claims the component properties. The certification issues have traditionally been of interest only to few domains but the increasing complexity of component-based systems has set them off as a widespread necessity.

6. Conclusions

Since its beginning in January 2000, CLARiFi has undertaken research, requirements capturing, design, development, and evaluation activities. The project has developed a working prototype that offers interfaces to suppliers and integrators, with a flexible classification scheme that can evolve as a consequence of feedback from users. The project has hosted two forums for the discussion of the research and development progresses. Interaction with the industry has been achieved through the commitment of the industrial partners, the involvement of component suppliers in the requirements capturing, and the participation of CBSE experts at the forums.

At the time of writing, the final demonstrator, after three prototypes, has been completed. The project finished in June 2002 after four formal reviews by the European Commission. Due to the complexity and the effort required for an exhaustive evaluation, this activity was only partially completed during the UE-founded period and it is still an ongoing activity. The final system has been adopted by one of the industrial partner as an internal tool.

The experience collected during the CLARiFi project could be applied to a different application domain—web services—that shows many similar problems in the integration process.

Acknowledgements

This work has been co-funded by the European Commission (CLARiFi project, IST-1999-11631). We would like to thank our project partners for their stimulating input.

References


Justin Clark, a graduate of 1993 from Bristol Business School, University of the West of England, joined BTexact Technologies in 1998. He managed a number of advanced research ‘proof of concept’ projects and was also involved in their down streaming into the current Business Intelligence portfolio. Projects included advanced taxonomy/ classification modelling and advanced visualisation techniques. His current role is Business Development manager for the Business Intelligence unit. Business Intelligence is concerned with enabling a business to work smarter and to mitigate the risk associated with change. To survive and thrive the modern business needs to anticipate and adapt quickly to many internal and external demands and opportunities. This places a significant burden on the key corporate decision makers who, now more than ever, need to be in control of strategic elements of information.

Chris Clarke joined BT after completing his BSc(ENG) Computing at Imperial College in 1985. He has gained his software engineering experience designing bespoke solutions for the EFTPOS market and a world-wide distributed service management product suite. His business experience was widened by his move into the R&D arm at Martlesham developing C7 Planning systems, QoS & access fibre supporting OSS process design. Latterly he has worked for BTexact Technologies addressing the challenges inherent to CBSE, re-use, configuration, and change management within major business re-engineering projects, and leading a development team working for the EU 5th framework CLARiFi consortium.

Stefano De Panfilis is a Doctor in laude in Mathematics from the University of Rome “La Sapienza”. He was employed as software engineer by Engineering since 1984. In April 1986 he became Project Manager. In this responsibility he managed a business project of some 30 person/years including several subcontractors. By July 1988 he joined the R&D department with the responsibility of the “Formal Methods” unit. Since 1993 he supported Engineering to receive the ISO-9001 compliance certificate. In April 1994 he became responsible of the Research Workpackage of the SQUID Project (ESPRIT). Since January 1996 he managed several EC funded research projects such as DOOR, VALSE, SQUAD CLARiFi, and CBSEnet. Since January 2003 he is the Manager of the R&D Laboratory. He is author of several publications both on conference proceedings and journals.

Giampiero Granatella studied Computer Engineering at the University of Genoa, Italy. His works are related to CBSE and Metrics for
Software Components. He participated at European project CLARiFi, and at present he is working in the Liguria Region participating in projects related to e-government.

**Paolo Predonzani** joined BTexact Technologies in 2001 as business analyst and delivery manager for business intelligence solutions. He has applied enterprise knowledge management techniques to provide customers with models and views over the disparate information sources available in a business environment. In his recent work, he has delivered solutions for integrating and classifying marketing information into flexible profiles. He holds a PhD in Electrical and Computer Engineering from the University of Genoa, Italy, where he developed techniques for selecting software components.

**Alberto Sillitti** is a PhD Student in Electrical and Computer Engineering at the University of Genoa, Italy, where he took a Laurea degree in 2001 (summa cum laude). In 2000 he developed a real-time simulator of an Italian highway, then joined the CLARiFi project in 2001. His research areas include software engineering, component-based software engineering, integration and measures of web services, and agile methodologies.

**Giancarlo Succi**, Ph.D., PEng is Professor of Software Engineering, Director of the Center for Applied Software Engineering and Associate Dean (Curriculum and Student Affairs) at the Free University of Bozen. His research areas include empirical software engineering, software product lines, software reuse, software engineering over the Internet. He is author of more than 100 papers published in international conferences and journals, and of one book.

**Tullio Vernazza** received a graduate in Electronic Engineering from the University of Genoa in 1971. He worked on the design of the central processor for a digital telephone exchange in Telettra from 1971 to 1974. Then he joined the Faculty of Engineering of Genoa University as Assistant Professor in 1974 and as Associate Professor in 1983. Actually he is holding the courses of Calcolatori Elettronici and Calcolatori Elettronici II at the third and fourth year of the Laurea Degree in Computer and Control Engineering, University of Genoa.