TOWARDS A COMPONENT AND SERVICE MARKETPLACE WITH BRECHÓ LIBRARY

Cláudia Werner[1], Leonardo Murta[2], Anderson Marinho[1], Rodrigo Santos[1], Marlon Silva[1]
[1]*COPPE/UFRJ – Federal University of Rio de Janeiro, P.O. Box 68511 – Rio de Janeiro/RJ 21945-970 – Brazil*
[2]*Institute of Computing – Fluminense Federal University, Niterói/RJ – Brazil*

ABSTRACT

Component and service libraries are important to meet cost-effectiveness and productivity goals in Software Reuse with Component- and Service-Based Software Engineering. These libraries need to provide diversified mechanisms to help the service and component management processes, exploring aspects such as organization, evolution, trading, and underlying markets. This is important to a broad definition of value for components and services. This paper presents Brechó, a Web-based library that provides publication, documentation, storage, search, and retrieval mechanisms for components and services. Apart from these main functionalities, Brechó has features and advanced mechanisms that distinguishes it from other component libraries, such as market analysis for pricing activities and automatic generation for services.

KEYWORDS

Reuse Libraries, Components, Services, Software Markets, Software Reuse.

1. INTRODUCTION

Considering researches in the Software Reuse field [Frakes and Kang, 2005], an approach, and probably the most popular, is Component-Based Software Engineering (CBSE) [Szyperski *et al.*, 2002]. CBSE consists in the use of technical procedures to design and code software components, producing systems from these components and building them in predictable environments [Szyperski, 2003]. In this sense, a CBSE process must focus both on the building of systems from reusable components (development *with* reuse) and on the reusable components building (development *for* reuse) [Moore and Bailin, 1991]. An important element in this scenario is the component library [Sametinger, 1997] which supports component reuse during the software life cycle, aiming to meet cost-effectiveness and productivity goals [Luqi and Guo, 1999]. The main rationale for its existence is to provide ready access to reusable components for development teams and maintenance organizations, and to support system composition and rapid prototyping [Mili *et al.*, 1994].

Despite the growth of component-based applications from 28% to 70% in the period of 1997 and 2002 [Yang *et al.*, 2005], there are some problems regarding the library organization and maintenance, and the generation of an industry and a market that explore CBSE challenges. The most critical problems underlie in difficulties in: (i) standardizing components (interface issues and distribution channels among supply and demand) [Overhage and Thomas, 2004]; (ii) providing useful information to improve decision making processes (quality information extracted from historical data, considering specific stakeholders) [Santos *et al.*, 2009]; and (iii) providing guidelines for making adaptation and intellectual property aspects [Brereton *et al.*, 2002]. Mature industrial application development is reuse-driven and uses existent artifacts where possible, contributing with improvements (time-to-market, maintainability, scalability) and maturing internal component repositories [Overhage and Thomas, 2004]. Moreover, a key prerequisite for taking advantage of these potential improvements and thus leveraging CBSE in general is to treat: the supply of components with good quality and easily found, understood, acquired and reused from Internet-based distribution channels [Bass *et al.*, 2000]; and the difficulties in inserting and realizing the value of components [Traas and Hillegersberg, 2000], due to the complexity in analyzing the facets in their definition (i.e., costs, benefits, risks, time, opportunity, necessities, flexibilities, behavior and requirements) [Santos *et al.*, 2009].

According to Szyperski *et al.* (2002), an imperfect technology in a working market is sustainable (through sociotechnical interactions) and a perfect technology without any market will vanish. Considering that there were many research works on component libraries (a lot of these not publically known) and search and

retrieval mechanisms in the 80's and 90's [Mili *et al.*, 1998], such markets can rarely be found in practice and the marketplaces that evolved up to now were unable to establish themselves and rapidly dropped out of sight again [Hahn and Turowski, 2003]. In parallel to this scenario, the CBSE maturity can leverage another emergent and potentially promising technique: Service-Based Software Engineering (SBSE) [Stojanovic and Dahanayake, 2005]. This technique is based on the services concept instead of components, and thus applications based on service-oriented architecture (SOA) can be developed. SOA solves some shortcomings of CBSE, especially interoperability (among platforms, devices, programming technologies, communication, middleware, and data format) [Stal, 2002]. This is due to the fact that services are not executed in the requester's machine, but in a remote provider [Papazoglou, 2003]. Therefore, there is no need to deploy components in specific platforms. Services also use standard languages and protocols, which are common in any information technology environment. Bearing in mind that services can be implemented concretely by components, joining new mechanisms to component libraries can be a strategy to enhance the way of exploring Software Reuse Libraries benefits, stakeholders' needs and libraries features.

Considering that one of the traditional issues for SBSE and CBSE's market growth is the development of a library that supports component and services lifecycle stages and management functions in different layers and takes into account all stakeholders activities and perspectives [Traas and Hillegersberg, 2000] [SOFTEX, 2007], this paper presents a component and service library named Brechó, which is a Web information system that provides publication, documentation, storage, search, and retrieval mechanisms. The focus of this paper is to describe the general approach that supports the Brechó Project as well as the relationship among its mechanisms, aiming to organize reusable assets in a distribution channel. Additionally, Brechó explores the congruence of stakeholders needs and provides advanced features that support service and component marketplace through (i) flexibility of the component concept and personalized packaging; (ii) flexible publishing mechanisms; (iii) version and evolution control of components; (iv) advanced search and retrieval mechanisms; (v) management of component producers and consumers; (vi) management of licenses and contracts during components retrieval; and (viii) mechanisms for service publishing, negotiation, and execution. The next section presents Brechó, describing its internal structure, and fundamental and advanced mechanisms. Section 3 reviews some related works. Finally, Section 4 concludes with some future directions.

2. BRECHÓ: COMPONENT AND SERVICE LIBRARY

As mentioned before, component and service libraries are important to meet cost-effectiveness and productivity goals in Software Reuse. Usually, they support four main functionalities related to component management processes [Brereton, 1999] [Luqi and Guo, 1999]: storage, publishing, search, and retrieval. Based on this context, the presented work was developed within the Brechó Project [Brechó, 2009]. The goal is to explore topics related to component and service libraries and marketplaces aiming to support fundamental and advanced mechanisms for a decision making process and a market environment with reusable assets. Section 2.1 presents Brechó's internal structure. Its fundamental mechanisms are described in Section 2.2. The following sections describe some Brechó's advanced mechanisms (Sections 2.3 to 2.5).

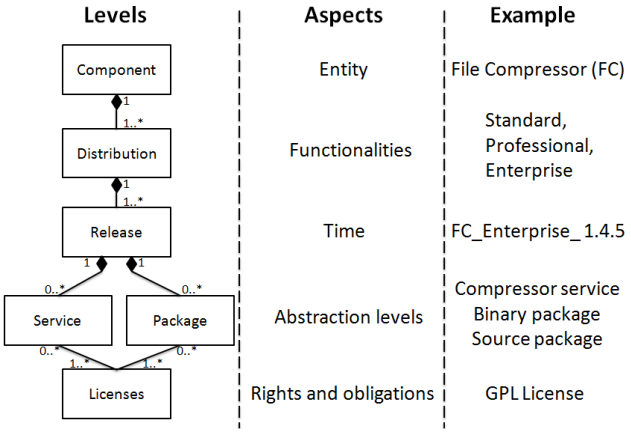
2.1 Internal Structure

Having the objective of better classifying and representing a component in a library, the internal structure of Brechó is layered into levels. Each level represents a component dimension, taking account of different aspects. Figure 1 illustrates these component levels showing a component example (*File Compressor*).

The first level, **Component**, conceptually represents the entities stored in Brechó. Concrete information about the implementation of these entities is not comprised at this level. According to Figure 1, one example of this level is the *File Compressor (FC)* component. The **Distribution** level represents the functional dimension, which classifies an entity in specific combinations of features that are desired for particular groups of users. Following the example in Figure 1, the FC component has *Standard*, *Professional*, and *Enterprise* distributions. The **Release** level is the time dimension, which defines the versions of artifacts that implement the entities in a specific moment in time. Each release is related to a specific distribution. For example, the distribution Enterprise has the release *FC_Enterprise_1.4.5*, as shown in Figure 1. From this level, the entities have concrete information about their implementations, and they are represented in

different levels of abstraction (e.g., analysis, design, code, binary etc.). Besides, different packaging can be set to enable the reuse of the available levels of abstraction (e.g., a package containing design and binary). Thus, the **Package** level represents a cut on the levels of abstraction, enabling artifacts to be set according to reuse consumer groups. Following the example of Figure 1, the release FC_Enterprise_1.4.5 has the *Binary* and *Source Package*. The **Service** level is a specific abstraction level in which the release to be used as services is enabled. In Figure 1, the compressor functionality is also provided by service. Finally, the **License** level allows the definition of rights and obligations on packages and services. Each package and service may provide several licenses, which guarantee rights and obligations between producers and consumers. One of these licenses must be chosen during reuse. In Figure 1, the license available is the *GPL license*.

Figure 1. Brechó's internal structure



2.2 Fundamental Mechanisms

Regarding the **storage** process, Brechó adopts a flexible concept of component that goes beyond a binary artifact, as was discussed in Section 2.1. The goal is to store all artifacts produced during the component lifecycle. Furthermore, services can also be stored in Brechó. These services can be provided by producers or automatically generated and hosted by Brechó (see Section 2.3). The components and services stored at Brechó are only visible to consumers when the producer decides to publish them. Therefore, Brechó can be used as a private library by producers to store their components under quality evaluation.

The component **publishing** process is divided in different phases at Brechó. This is due to its internal structure, which classifies a component in different aspects. From this organization, it is possible to publish several component and service releases for a unique component. In addition, the component documentation structure is based on categories and customizable dynamic templates. Each category has documentation templates, and a component inherits these documentation templates for each category to which it is associated. For example, if a component is associated to a category named *Java*, it inherits a documentation template that requires information about Java components (e.g., which Java distribution the component uses: Java SE, Java EE, or Java ME). Therefore, it is possible to classify a component in various categories and associate them to different documentation templates. This component documentation resembles a mosaic.

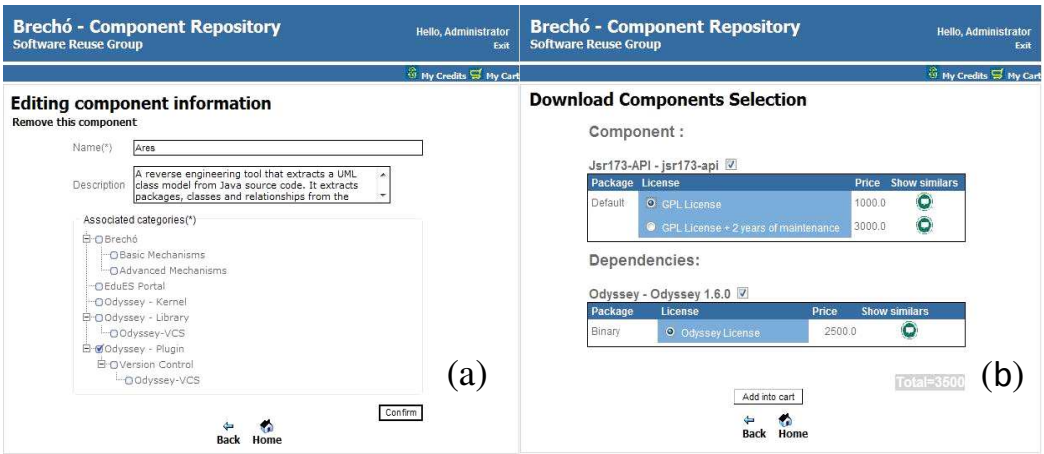
The **categorization** mechanism supports hierarchical categories using concepts related to directed acyclic graphs (DAG). In this mechanism, there are *super* and *subcategories* (similar to classes in object-oriented development), in two situations, depending on the classification process adopted at Brechó: a (sub) category can belong to more than one (super) category (uniquely named), or the mechanism can work similar to directories in file system, where categories with the same name in different hierarchies are distinct categories. Also, the mechanism supports two views: *horizontal* (linear, browsing view of hierarchical categories, considering actions related to categories maintenance) and *vertical* (tree, global view of hierarchical categories, considering actions related to components maintenance, as shown in Figure 2.a). This mechanism can contribute to search and retrieval activities because it improves classification strategies in component libraries and stakeholder decisions (e.g., search and retrieval activities by consumers), trying to support different ways to get a required component or service. Furthermore, the component categorization at Brechó works dynamically, and producers can participate, suggesting new categories. This process can be manual

(producer inserts expressions in text fields) or automated (Brechó searches for suggestions using Web engines). Moreover, the library manager can check the most important suggestions comparing the producer suggestions to existent categories in Brechó in order to filter and organize results.

In the component **search**, Brechó goes beyond basic mechanisms, such as keyword search, enabling component consumers easily and quickly to find what they want. From category browsing mechanisms, consumers can narrow their search by navigating through categories. The search can also be narrowed via the specification of component documentation values using the filtering mechanism. For example, a user may filter components from category *Java* that are classified as *Java SE*. The keyword search mechanism considers *synonyms* and *similarity* between words and checks all *sources of documentation* associated to the component (through the categories to which it belongs). Therefore, if the user types a wrong keyword, Brechó suggests a list of possible correct keywords to the user.

Considering the **retrieval** mechanism, Brechó works with the concept of *purchase cart*. Consumers add components in the cart while they are navigating through Brechó. When consumers add a component into their cart, they must choose one of the licenses provided by the component. Moreover, Brechó also presents to consumers the *component dependencies* (Figure 2.b), where a release of a component can depend on releases of other components, considering transitive relationships in order to make the publishing (producer) and purchase (consumer) process aspects more explicit. This kind of mechanism is important to guarantee that the consumers are aware of all requirements needed to use the component adequately. In addition, it eases the component acquisition process, since it saves the consumers' time from searching for additional components. However, the component dependency information is not automatically generated. It must be previously configured by producers when they publish their components. To finalize the retrieval, consumers must be authenticated in Brechó. In this moment, Brechó creates a contract between producer and consumer, transferring credits from the consumer account to the producer account (see Section 2.4 and Section 2.5).

Figure 2. Component editing and retrieval screens



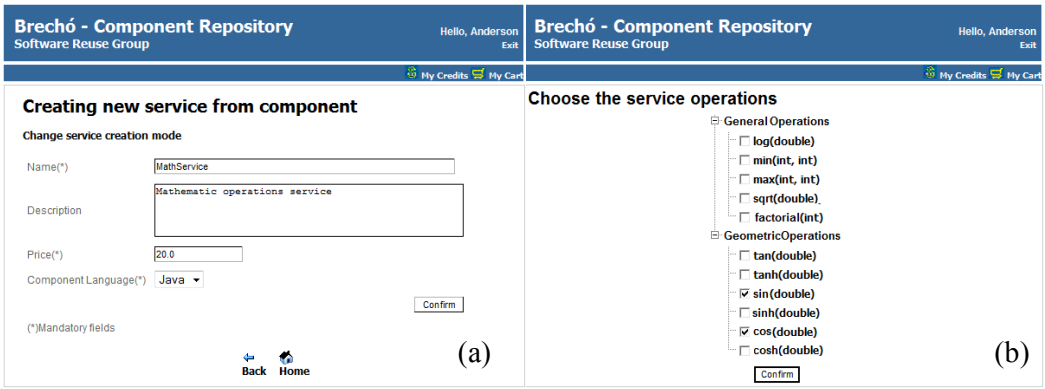
2.3 Service Publishing

Brechó provides three strategies for publishing **services**: external services, for already existing services; hosted internal services, for services generated from previously published components in the library; and hosted composite services, for services constructed from the composition of external and internal services [Marinho *et al.*, 2009]. The first strategy assumes that producers already have services running in a server and they want to publish them in Brechó. The second and third strategies assume that producers do not have resources to provide services and request the repository to provide them, publishing in the library. In the second strategy, however, services are generated from components, which producers have previously published in Brechó, and hosted by the library. While in the third strategy, composite services are constructed from BPEL files written by producers. These services are then hosted and orchestrated by Brechó.

Figure 3 shows the internal services creation at Brechó. In order to create an internal service, the producers should choose the component they want to expose as service. Actually, the producers select a release of a component, since component artifacts, such as component's source and binary code, are located

at this level. After that, the producers should fill in a form, which requires some information about the internal service that is going to be created, as shown in Figure 3.a. This information is: name, description and the programming language in which the component has been coded. This last information is needed to allow the selection of the appropriate mechanism to generate the service. It is important to highlight that Brechó currently provides the mechanism responsible for generating services from Java components; however, other mechanisms like that can be easily integrated to Brechó's architecture. When the producer submits the form, the available operations for the service are shown (Figure 3.b), extracted from a specific release of the component in which the producer previously chose to publish the service. In this example, a service is being generated from a component coded in Java. The available operations are shown in a *tree form*, where *nodes* represent *classes* and *leaves* represent *methods* (the operations to choose from). According to Figure 3.b, the *sine* and *cosine* will be transformed into services.

Figure 3. Service editing and extracting screens



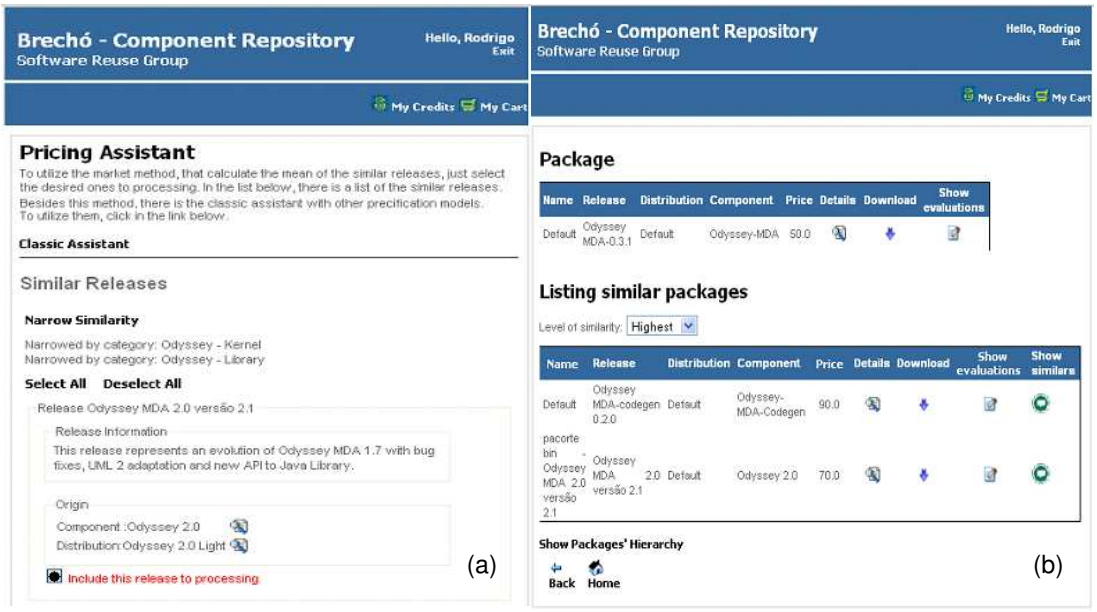
2.4 Financial and Pricing Mechanisms

At Brechó, there is a financial mechanism responsible for supporting different ways of **pricing** reusable assets, according to the acquisition or use of concrete artifacts or services from published components, respectively. The process of charging services is possible because all services published in Brechó are under an access control mechanism. The *financial* mechanism implemented at Brechó combines pre and postpaid models from the mobile telecom domain, and some concepts from the bank domain, where users have a credit limit to spend with purchases, but they can buy credits to compensate the debit (postpaid model), or to accumulate credits (prepaid model). This information is set in each user account by the library manager.

The financial mechanism is composed by a *pricing* mechanism which aims to analyze how the value can be realized by stakeholders in a quantified way, considering different perspectives (producers and consumers) and a value chain instance (pricing and purchase) [Santos *et al.*, 2009]. Pricing activity occurs when a producer is uploading artifacts of a release that belongs to a specific distribution of a component. Due to the concept of credits, Brechó supports collaborative and competitive scenarios and it helps to determine the price of components considering information extracted from producers and consumers actions. When producers (beginners or experts) are publishing a new component release, they determine the prices of its artifacts (e.g., documentation, source code, binary etc.). However, these artifacts can have similar ones or not (similar or innovative). Thus, two pricing methods are provided to producers: *classic* (financial features of artifacts must be set by producers in order to get a diagnosis from equations of Economy) or *market-based* (similar artifacts are shown to producers from *market* similarity based on *syntactic* and *historical* similarities, and they can mark real similar artifacts – Figure 4.a – to execute a market analysis aiming to check similar artifacts' mean price, evaluation, and number of consumers). Considering consumers perspective, similar artifacts (Figure 4.b) marked by producers during pricing activities (depending on reputation and experience) can improve search and retrieval using producers feedback (similarity based on collective intelligence [Segaran, 2007]) and markets movements [SOFTEX, 2007] to enhance the consumers' realization of value.

After pricing artifacts, producers create packages and services over them and, in this moment, Brechó makes automated calculations. For example, if the producer creates a package *W* from an artifact *A* priced $1000 and an artifact *B* priced $5000, selecting a license *C* that costs $2000, the inferred final price is $8000.

Figure 4. Component pricing and purchase screens



2.5 Reuse Map

Brechó provides the functionality called *reuse map* [Lopes *et al.*, 2006], in charge of identifying maintenance responsibilities. The reuse map keeps information about contracts between producers and consumers, for each reused component or service. It also has a registry of licenses to which each contract is associated. The licenses define, in general terms, the rights and obligations on the use of a component or a service, creating a channel between producers and consumers. This channel allows the notification of updates, demands for changes, bug fixes, and inclusion of new functionalities, supporting the component evolution.

From the reuse map information, teams involved in the maintenance process can determine who is responsible for changing a specific component, by identifying the producer and the contract terms related to that component. In addition, it is possible to get the component's consumers list with the objective of informing them about a new component release. Therefore, consumers can select their last purchases and producers can verify the amount of consumers considering a specific component. Besides supporting maintenance, the reuse map can be extremely useful to help producers and consumers to make good deals in Brechó. For instance, a producer can adjust the component or service prices according to the usage frequency. On the other hand, a consumer can use this information as a reliability parameter of a component or service. This is possible due to the *evaluation* mechanism that allows consumers to provide satisfaction feedback about a component retrieved from Brechó. This feedback occurs with an attribution of a satisfaction degree (from bad to excellent) and textual comments (suggestions, praises, complaints etc.). Also, this information can be visualized (for each reusable asset) by producer and consumers through a graphical chart.

3. RELATED WORK

Brereton *et al.* (2002) developed an infrastructure to support the practical application of CBSE in a marketplace, called *CLARiFi*, considering the results described in [Traas and Hillegersberg, 2000]. It distinguishes three roles (supplier, integrator and broker, similar to Brechó) and foresees the presence of a wide selection of COTS (Commercial-of-the-Shelf) components. Moreover, *CLARiFi* establishes models to improve component classification, certification, and conformance to standards, ranking and selection, and visualization, based on the identification of some properties of components required from suppliers to maintain the library organization quality. Different from Brechó, *CLARiFi* focuses on search and retrieval activities, doing these in a broad way and trying to follow an automatic approach. On the other hand, Brechó

explores a set of various and connected mechanisms that are semi-automatic, in order to link with the underlying reuse process, besides treating SBSE and a market scenario. Overhage and Thomas (2004) investigate the lack of a strong component market and its reasons, and propose a marketplace model, *CompoNex*, for trading software components that is currently being developed to face difficulties and limitations in component trading and to facilitate component exchange by buyers and sellers under these conditions. They build their work over the economic theory of perfect markets and criticize sparsely populated market segments, considered as a dominant market power, and the way to treat information about components. They explore a specification framework and related classifications in order to actively match supply and demand. Once again, Brechó presents a different behavior when compared to *CompoNex* because Brechó provides more advanced mechanisms distinct from classification and documentation, such as SBSE, evaluation, pricing, hierarchical categorization, level-based internal organization etc.

In the Web, there are many component libraries that are available for use [ComponentSource, 2009] [JIDE, 2009] [Xtras, 2009] and are characterized by cataloging of COTS components. Specially, ComponentSource has more than one million of registered users and more than one thousand published components, being the most active marketplace for fourteen years. Although ComponentSource explores marketing strategies to maintain a Web store, it does not support pricing activities nor the visualization of information extracted from historical data. Additionally, Logidex (2009) and Select Asset Manager (2009) are component business libraries that keep information about reusable components used in an organization. They store information about component dependencies and utilization. However, these approaches focus on exploring and combining existent mechanisms related to search, classification, and retrieval, lacking service publishing and pricing mechanisms.

4. CONCLUSION

Component and service libraries are important to meet cost-effectiveness and productivity goals in Software Reuse with CBSE and SBSE. Libraries need to provide diversified mechanisms to help service and component management processes in order to contribute to the organization, evolution and trading in libraries and underlying markets, exploring definition of value for components and services. Thus, this paper presented Brechó, a library which establishes a channel between producers and consumers, allowing components and services to be cataloged and retrieved. Brechó aims to contribute with CBSE and SBSE scenarios by providing an Internet-based distribution channel that allows easy integration with activities and results of Reuse Management Processes, and an approach to support a marketplace focus on improving decision making processes considering stakeholders' proposition and realization of value. Brechó is supported by the following mechanisms and features: (i) a well-defined component internal structure layered in levels; (ii) a mechanism for mapping producers and consumers relationships (reuse map); (iii) a mechanism for component evaluation; (iv) a mechanism for services generation from components; (v) a mechanism for pricing components and services; and (vi) mechanisms for expanding a library to generate a value-based component and service marketplace.

New mechanisms are being developed (and evolved) to expand the financial and pricing mechanisms, considering a set of value facets beyond costs, such as *marketing*, *evaluation*, and *negotiation* mechanisms. The goal is to explore historical data and information visualization produced from stakeholder actions in a value-based marketplace aiming to collect useful data, using Value-Based Software Engineering concepts [Biffl *et al.*, 2006]. These data are fundamental to make new classic models that really describe possible and trustable components and services' behavior, linking CBSE/SBSE to Software Economics [Boehm and Sullivan, 2000]. Additionally, Brechó has been used at Software Engineering Laboratories in academic scenario for the last two years. However, experimental studies with software engineers using Brechó are being planned aiming to verify implemented mechanisms in industry, mainly case studies in real scenarios due to the main motivation: to provide component and service markets. Studies related to definition of value for services can be explored as well. In this way, this research can contribute to explore new strategies to make component markets an important step towards exposing CBSE and SBSE to new challenges.

Acknowledgments. The authors would like to thank CAPES, CNPq and FAPERJ for the financial support, and to all participants of Brechó Project.

REFERENCES

Bass, L. *et al.*, 2000. *Market Assessment of Component-Based Software Engineering*. Technical Report CMU/SEI-2001-TN-007, Software Engineering Institute, 33p.

Biffl, S. *et al.*, 2005. *Value-Based Software Engineering*, Springer-Verlag.

Boehm, B. and Sullivan, K., 2000. Software Economics: A Roadmap. *Proceedings of the Conference on The Future of Software Engineering*, 22nd International Conference on Software Engineering, Limerick, Ireland, pp. 319-343.

Brechó, 2009. *Brechó Project*, http://reuse.cos.ufrj.br/brechoproject.

Brereton, P., 1999. Evolution of Component Based Systems. *Proceedings of the International Workshop on Component-Based Software Engineering*, Los Angeles, USA, pp. 123-126.

Brereton, P. *et al.*, 2002. Software Components – Enabling a Mass Market. *Proceedings of the 10th International Workshop on Software Technology and Engineering Practice*, Washington, USA, pp. 169-176.

ComponentSource, 2009. *ComponentSource*, http://www.componentsource.com.

Frakes, W. B. and Kang, K., 2005. Software Reuse Research: Status and Future. *IEEE Transactions on Software Engineering*, v. 31, n. 7 (July), pp. 529-536.

Hahn, H. and Turowski, K., 2003. Drivers and Inhibitors to the Development of a Software Component Industry. *Proceedings of the 29th EUROMICRO Conference*, Antalya, Turkey, pp. 128-135.

JIDE, 2009. *JIDE*, http://www.jidesoft.com/.

Logidex, 2009. *Logic Library Logidex* , http://www.logiclibrary.com.

Lopes, L. *et al.*, 2006. Odyssey-CCS: A Change Control System Tailored to Software Reuse. *Proceedings of the 9th International Conference on Software Reuse*, Torino, Italy, pp. 170-183.

Luqi and Guo, J., 1999. Toward Automated Retrieval for a Software Component Repository. *Proceedings of the Conference and Workshop on Engineering of Computer-Based Systems*, Nashville, USA, pp. 99-105.

Marinho, A. *et al.*, 2009. Extending a Software Component Repository to Provide Services. *Proceedings of the 11th International Conference on Software Reuse*, Falls Church, USA, pp. 258-268.

Mili, A. *et al.*, 1994. Storing and Retrieving Software Components: A Refinement Based System. *Proceedings of the 16th International Conference on Software Engineering*, Sorrento, Italy, pp. 91-100.

Mili, A. *et al.*, 1998. A Survey of Software Reuse Libraries. *Annals of Software Engineering*, v. 5, January, pp. 349-414.

Moore, J. and Bailin, S., 1991. Domain Analysis: Framework for Reuse. In: *Domain Analysis and Software System Modeling*, Los Alamitos, USA, pp. 179-203.

Overhage, S. and Thomas, P., 2004. A Business Perspective on Component Trading: Criteria, Immaturities, and Critical Success Factors. *Proceedings of the 30th EUROMICRO Conference*, Rennes, France, pp. 108-117.

Papazoglou, M., 2003. Service-Oriented Computing: Concepts, Characteristics and Directions. *Proceedings of the Fourth International Conference on Web Information Systems Engineering*, Rome, Italy, pp. 3-12.

Sametinger, J., 2001. *Software Engineering with Reusable Components*, Springer-Verlag, USA.

Santos, R. P. *et al.*, 2009. Incorporating Information of Value in a Component Repository to Support a Component Marketplace Infrastructure. *Proceedings of the 10th IEEE International Conference on Information Reuse and Integration*, Las Vegas, USA, pp. 266-271.

Segaran, T., 2007. *Programming Collective Intelligence*. O'Reilly, 1. ed., USA.

Select Asset Manager, 2009. *Select Asset*, http://www.selectbs.com/adt/software-asset-management/select-asset-manager.

SOFTEX, 2007. *Perspectives on Development and Use of Components in a Software and Services Brazilian Industry*. Technical Report (in Portuguese), SOFTEX-MCT-DPCT/Unicamp, Ministry of Science and Technology, 40p.

Stal, M., 2002. Web Services: Beyond Component-Based Computing. *Communications of the ACM*, v. 45, n. 10 (October), pp. 71-76.

Stojanovic, Z. and Dahanayake, A., 2005. *Service-Oriented Software System Engineering Challenges and Practices*. IGI Global Publishing, Hershey, USA.

Szyperski, C. *et al.*, 2002. *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley and ACM Press, 2. ed., Boston, USA.

Szyperski, C., 2003. Component Technology: What, Where, and How?. *Proceedings of the 25th International Conference on Software Engineering*, Portland, Oregon, pp. 684-693.

Traas, V. and Hillegersberg, J. V., 2000. The Software Component Market on the Internet: Current Status and Conditions for Growth. *ACM SIGSOFT Software Engineering Notes*, v. 25, n. 1 (January), pp. 114-117.

Xtras, 2009. *Xtras*, http://xtras.net/.

Yang, Y. *et al.*, 2005. Value-Based Processes for COTS-Based Applications. *IEEE Software,* v. 22, n. 4 (July-August), pp. 54-62.