

Open issues and concerns on Component Based Software Engineering

Stefano De Panfilis² and Arne J. Berre¹

²Engineering Ingegneria Informatica S.p.A.
Via San Martino della Battaglia, 56
00185 Roma – Italy
stefano.depanfilis@eng.it

¹SINTEF, Forskningsveien 1, Blindern, 0314 OSLO, NORWAY
Arne.J.Berre@sintef.no

Abstract

This paper summarises two years of analyses conducted during the *Component Based Software Engineering network* (CBSEnet) work by a panel of several experts. The summary shows to practitioners issues in adopting Component-Based Software Engineering (CBSE) practices within industrial organisations and to researchers still open issues which requires further investigations in the coming years.

Introduction

Software components are building blocks of software. In today's world, a software component is any piece of pre-written code, with defined interfaces, that can be called to provide the functionality that the component encapsulates. These are typically packaged in "industry standard" ways so that they can be callable from multiple languages, or from multiple environments. Typically components are created as Microsoft® .NET or Microsoft Component Object Model™ (COM) components, Java™ 2 Platform Enterprise Edition (J2EE) or JavaBeans™ components, Borland Delphi™ VCLs, or a number of other lesser known architectures. The new paradigm of assembling components and writing code to make these components work together has a name, and of course an acronym, Component-Based Development (CBD), while the whole discipline including components identification, development, adoption and integration in larger software systems is called Component-Based Software Engineering (CBSE).

The objective of this document is to survey results discussed in a detailed way within the "CBSE Landscape Document", one of the key results of the CBSEnet project. In particular the Landscape Document defines key research issues and requirements for CBSE in a number of different application domains. Economic, managerial and organisational issues are taken into account. The domain-specific nature of the landscape document distinguishes it from other surveys of this type and ensures that the research identified is associated with problems in a specific application domain.

A domain-specific document is required partly because CBSE already means different things in different application domains and partly because the specific nature of different types of application place different requirements on CBSE. For example, there is a radical difference between CBSE for information systems and CBSE for large, long-lifetime command and control systems. However, we also recognise that certain critical issues will cut across domains. These horizontal issues will play a crucial role in the document.

To this end we organised issues into vertical and horizontal structures to take into account domain specific and cross-domain issues. Vertical structures are associated with application domains and reflect domain specific aspects. Horizontal structures reflect aspects that cut across domains such as process, technology and related development paradigms.

This document, being a short version of the more complex Landscape Document, is organized in chapters each of which, regardless is addressing an horizontal or a vertical area, identifies a priority research topic, and is structured as follows:

- Research topic short definition
- Why research is needed in this topic (rationale and justification)
- Research needed in this topic (list open issues with short explanation)

The list of research areas identified are:

- Business information systems
- Geographical Information Systems
- Embedded Systems
- Finance Systems
- Telecommunication Systems
- CBSE Development Processes
- Commercial of the shelf components
- CBSE and related development paradigms
- CBSE Product issues

Within these we have structured the discussion of research topics from the different domains above in the following groups – related to the relevant solution approaches, such as CBSE, Service-oriented architectures, Model-driven architecture, Aspect-oriented computing, etc. As CBSEnet we call these **CBSE-related development paradigms**. The groups are:

- Concepts
- Process
- Business
- Product
- Technology – Development environments / tools / Standards
- Off-the-shelf components/models
- Related paradigms

1. Domain-specific issues: Consolidated Common Issues

Analysing in general CBSE it is possible to identify a set of challenges which have to be addressed in order to put in place sound CBSE practices. Nevertheless those practices have different relevance for different application domains. The main purpose of the analysis performed within each specific application domain is to identify, driven by the set of identified challenges, the main gaps between the desired ideal situation and the current state of the practice. When appropriate, for each application domain, the list of common issues has been enriched by adding new specific domain challenges.

Each challenge is then discussed in terms of its relevance to the domain and then prioritized within an expected time-frame. The list of challenges and the main gap identified (Research, Technology, Standard), organised according to the CBSE Classification Model, currently is:

			GAP: (R,T,S)
CBSE Aspect	Concepts	Composition of components, choreography and orchestration	R,T
		QoS supporting infrastructure	R, T
		Evolution (process) – change management	R
		Specification of components (process)	R
		Handle multiple technologies	R,T
	Process	More well-documented CBSE, MDE, SOC engineering methods	R
		Component-development process and organisation issues	R,T
		Component evaluation, testing, validation	R
		Subcontracting	T
	Business	Domain components (standard)	S
		Intellectual property rights, licensing and marketing	T
		Fostering of market places (availability paradox)	R,T,S
		Availability of business case evidence	T
	Product	Certified components	R,S
		Trusted components	R
		Security and payment	T
		Extra functional aspects	R
		Quality of service	R
		Quality of data (content)	R/T
		Dependability Reliability	R
Adaptability, flexibility, self configuration		R	
Test/Validation/Verification		T	
Composition(ality)		R	
Security	T		
CBSE Aspect	Technology	Open Development environments and tools	R,S,T
		Standards – for services/information in domains	S
		Verification and test tools	T
	Off-the-shelf component	Identification, classification and characterisation	R
		Selection	T
		Repository	T
		Testing – self-testing components/services	R/T
	Related paradigms	Procurement with Trust	T
		Agent-based development	R
		Aspect-oriented development	R
		Model-driven development	R
		Service-oriented software engineering	R

2. CBSE Concepts

2.1 Research topic definition

The area of concepts deals with the principle foundation of CBSE technology, and its formal relationship to related technologies like Service Oriented Computing and Model Driven Engineering.

2.2 Why research is needed in this topic

The foundation and conceptual model for CBSE and related technologies like Service oriented computing and Model driven Engineering have some relationships when it comes to areas that requires more research.

2.3 Evidence of research need

Many people do not yet see clearly the relationships between technologies like objects, components, services, agents and the mode-based descriptions, - and to some extent view these as alternative or competing technologies. In particular when it comes to common areas across these technologies, like object aggregation versus component composition versus service choreography and orchestration there are more similarities than differences.

2.4 Research needed in this topic

Concepts	Composition of components, choreography and orchestration	R,T
	QoS supporting infrastructure	R, T
	Evolution (process) – change management	R
	Specification of components (process) !)	R
	Handle multiple technologies	R,T

3. CBSE and Processes

3.1 Research topic definition

Component-based software development is being proposed as a means of reducing costs while accelerating software development. The drive to use components to construct software systems stems from a ‘parts’ philosophy derived from traditional engineering disciplines that promises instant productivity gains, accelerated time to market and lower development costs. However, software component technology is still immature and poses many problems for organisations intending to adopt it.

To facilitate the adoption of CBSE practices, this research area addresses how organisations have to modify their development process in order to comprise CBSE. In particular two different processes have been identified: development **for** reuse, and development **with** reuse. In this area CBSEnet concentrates on the description of the processes for developing component-based applications from black-box off-the-shelf software components, and discusses the challenges associated with each development stage.

3.2 Why research is needed in this topic

Component-based system development proceeds by composing software systems from reusable components (often black-box third-party software) which supported features vary greatly in quality and complexity. Application contexts in which the components have been implemented also vary considerably. This complexity together with the variability in application domains means that specifications delivered with software components are likely to be incomplete or inadequate. This implies that software components are generally not “plug-and-play” and may require significant adaptation effort to use them in new situations.

These problems underpin the need for software engineering processes that can balance aspects of system requirements, business and project concerns, with the assumptions and capabilities embodied in off-the-shelf software components.

3.3 Research needed in this topic

Process	More well-documented CBSE, MDE, SOC engineering methods	R
----------------	---	---

	Component-development process and organisation issues	R,T
	Component evaluation, testing, validation	R
	Subcontracting (business+++)	

4. CBSE and Business

4.1 Research topic definition

This area deals with market and business aspects of CBSE with a particular concerns to IPR issues which immediately turns towards contractual agreements. Indeed, one of the main concern of software component adopters is the clear definition of the responsibility chain, i.e. from the component supplier to the end user via the application builder (this case the components integrator) who is the responsible when the service outsourced by the application doesn't work as expected.

Similar concerns exist when thinking on the intimate evolutionary nature of the current software systems.

4.2 Why research is needed in this topic

It is quite clear that the earlier promoted component market has not materialised as forecasted. There are many reasons for this, but further research is needed in this, to perhaps find ideas for successful approaches in the future.

4.3 Research needed in this topic

Business	Domain components (standard)	S
	Intellectual property rights, licensing and marketing	T
	Fostering of market places (availability paradox)	R, T, S
	Availability of business case evidence	T
	Certified components	R, S

5. CBSE and Product

5.1 Research topic definition

One of the main expected benefits resulting from the CBSE approach is to improve the overall software products quality. This area concerns how CBSE-based products properties are related with Software Quality issues such as those described in standards such as ISO-9126 and their following evolutions. This should be achieved by analysing each quality characteristic and its relevance with respect to CBSE trying to establish which of those should be addressed when applying sound CBSE practices.

5.2 Why research is needed in this topic

One of the crucial issue, also discussed since the first CBSEnet workshop, is how software components users could trust on external purchased software. This research branch is called "CBSE and trustworthiness". At present the way to "measure" at design-time and then at run-time the adherence of software components to their expected behaviour is rather unclear. Different approaches relies on results coming from other disciplines such as Software Architecture founding their solutions to specific ADLs (Architecture Description Language).

Although the approach is interesting no effective results for the final user of software components (system integrators) have been produced so far. The issue is even more complex when components are COTS.

5.3 Research needed in this topic

Product	Trusted components	R
	Security and payment	T
	Extra functional aspects	R
	Quality of service	R
	Quality of data (content)	R/T
	Dependability Reliability	R
	Adaptability, flexibility, self configuration	R
	Test/Validation/Verification	T
	Composition(ality)	R
	Security	T

6. CBSE and Reuse (COTS)

6.1 Research topic definition

In the early 90's a new type of software component came into existence. These were the commercially available components that could be purchased. ComponentSource coined a new term to describe these types of components: Open Market Components. Open Market Components are reusable software components that are available for purchase off the shelf. A lot of these components are based on standard component architecture such as COM, or Java and can be purchased without having to buy as well support, integration, or other types of services and they are sold as genuinely plug-n-play components.

There are a plethora of open market components available, over 1,000 at this writing, to make a programmer's job easier and at the same time, allow them to concentrate on programming their core competency tasks by implementing their corporation defined business processes or functionality instead of having to write all sorts of components or routines to do common things like data display, charting, calculations, algorithms, and many others that are available on the open market. Reuse has become a reality because programmers are able to reuse open market components, which is really just code that someone else has written, tested, and documented.

In addition to the term Open Market Component, there are many other terms have very similar meaning and basically refer to the same thing with a slight twist. One of the most commonly used term is commercial off-the-shelf (COTS) software products, most commonly referred to as COTS. Other terms such as off-the-shelf (OTS), and Software of Unknown Pedigree (SOUP) are also commonly used.

6.2 Why research is needed in this topic

The use of commercial off-the-shelf (COTS) software components is becoming an economic and strategic necessity for many organisations in a wide variety of different application areas including finance, defence, medicine, logistics, administration, manufacturing, and commerce.

However, employing COTS components in building applications is not a painless business. As the COTS software market develops, COTS users must face new challenges to successfully and effectively integrate commercial software components in applications and systems. COTS components are commercial software products developed for the open market and, therefore, they are not designed to meet the specific functional and integration requirements of a given organisation's applications. This simple fact has important implications in development and management activities of the COTS user organisation.

6.3 Research needed in this topic

Off-the-shelf component	Identification, classification and characterisation	R, T, S
	Process and Methods to manage and execute COTSs-based development projects	R, S
	Legal issues related to contractual aspects of COTSs acquisition	Regulation
	Cost Analysis. Aspects related to the evaluation of the economic feasibility of using a certain COTS against other options	R
	Change management. Managing and controlling the different types of change that may happen in COST-based development projects	S
	Repository	T

7. CBSE and related development paradigms

7.1 Research topic definition

The majority of research and development in CBSE has focused on the development and use of components within two development paradigms:

- ⌘ A RAD (rapid application development) paradigm where visual tools such as interface builders and form designers are used to create the user interface to an application and components are associated with elements identified in the interface. Microsoft, through Visual Basic and Visual C++, have been the principal proponents of this approach and its success has been largely due to the extensive libraries of COM components that are available. This approach uses an iterative approach to development and is particularly suitable for the development of small to medium sized business systems.
- ⌘ A ‘design-driven’ paradigm where the software is developed using a ‘conventional’ software life cycle. A software design is developed from a specification and this design is programmed in an object-oriented programming language (normally Java). Design notations such as the UML may be used. This approach may involve variable amounts of iteration and is most suited to the development of medium to large systems with demanding performance or dependability requirements.

Of course, there is considerable overlap between these approaches to development with CASE tools and program generators widely used in the ‘design-driven’ paradigm and significant component development in the RAD paradigm.

This research area is concerned to assessing the position of CBSE with respect to other development paradigms that are widely used for systems development and to discussing CBSE with respect to different software development processes in order to identify mutual beneficial impacts.

8. Conclusions

CBSE is a newly developed discipline which deserves several effective and perceived benefits, nevertheless its adoption requires, as any new practice, same care by software development organisations. The adoption concerns by practitioners turn easily into a variety of research issues still to be tackled and properly addressed. In particular to make effective the adoption of CBSE its differences towards other software development paradigms need to be made more clear and evident outside the market.

9. Acknowledgement

The authors wish to tanks the many experts who made this effort possible and their colleagues of CBSEnet. CBSEnet is totally funded by the European Commission under grant IST-2001-35485, See www.cbse.net