

TM

Softonomy

The Emergence of Components

6th January 2002

© Copyright 2002, Softonomy Ltd.

Softonomy Ltd

Growcorp Innovation Centre, 3015 Lake Drive, Citywest, Dublin 24, Ireland.

For queries or further information, you can reach us by:

Email: info@softonomy.ie

Telephone: +353 (0)86 821 0917

Fax: +353 (0)1 284 6381

Our website is located at: www.softonomy.ie

Company Registered in Ireland, No: 343729.

1. Introduction

Softonomy  are a young software technology company headquartered in Ireland. We provide an innovative **Software Development Service** that produces effective software solutions for Business Clients. What we provide for our Clients is really quite straightforward - we build *tailored* software application solutions that add value to a Clients business processes and their associated operational and strategic information systems.

The software application solutions we develop with our Clients use pre-built business process and software components, wherever possible. Typical benefits of our software solutions include:

- **Speed** - improved capture, handling, processing and management of information
- **Optimum Fit** – catering for specific business and informational needs
- **Efficiency** - the generation of efficiencies within internal and external business processes

This white paper outlines what is meant by a “component-based software paradigm” and how **Softonomy** uses this to beneficially develop software application solutions for our Clients.

2. Basis of our Research

Over the last two decades, we have investigated the vast majority of software development techniques, methodologies, programming languages, architectures and frameworks as well as information system and software development paradigms that have been proposed, developed and, if successfully maturing, worked on by both academic and commercial entities.

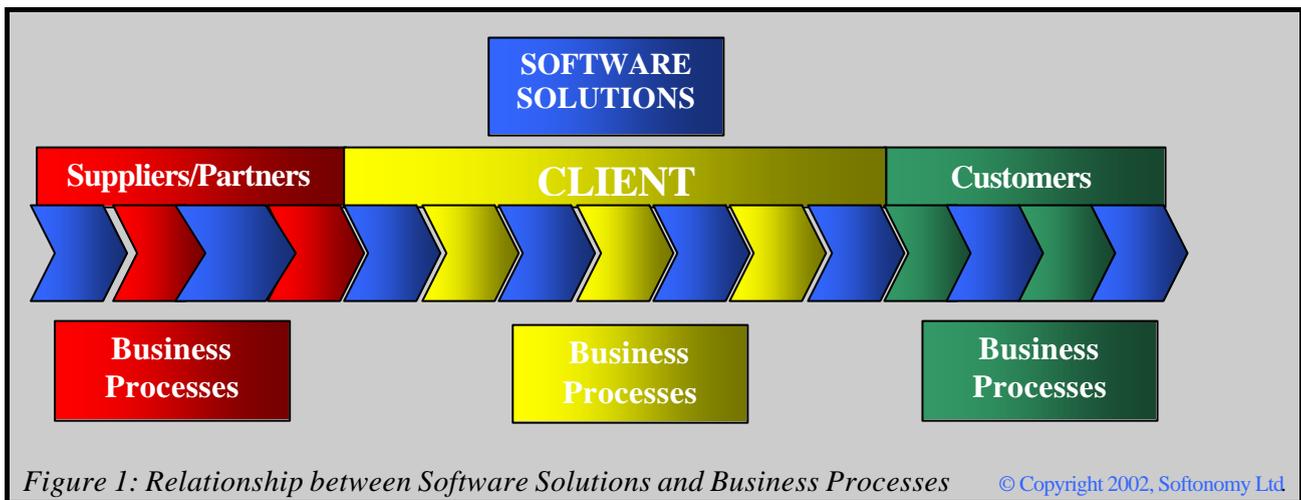
Additionally, we have been involved in developing real-world, robust and functionally complex software applications for major Clients. Many of these software applications have been supplied across an international customer base and have been subjected to demanding user and business environments. We've also developed smaller applications for less complex businesses, but nonetheless equally demanding Clients in terms of expectations and quality requirements. The Clients we have worked with have spanned across a wide range of industry sectors, such as Financial Services, Telecoms and Hi-Tech, to name but a few. Our software development experiences have helped us to understand what works and what doesn't.

Furthermore, in the last 18 months, we have enhanced our understanding of software development and its linkage to business benefit by investing in a research programme. This research programme consisted of detailed and extensive secondary desk research complemented by a quantity of basic primary research. This has added to our weight of knowledge of the software development area, and has allowed us to be convinced what are the current best practices and where and how they can be applied.

3. Business Opportunity

Our extensive research has shown that there is still very much to automate within businesses. For example, in Mar 2001, the *American Management Association* reported that one of the most important skills businesses need today is the ability to use information to address business challenges. Having the right information at the right granularity delivered to the right person and at the right time is key. Our Clients can use this as a **business opportunity**.

Clients have an opportunity to create “better” automation and information system “intelligence”, to increase business process effectiveness and to deliver efficiencies (see Figure 1).



4. Components

4.1 Background to Components

The idea that a software provider can provide solutions via the re-use of components is not new. Ever since “shared software libraries” were first put forward by *Doug McIlroy* in 1968, and then popularly re-iterated with “object integrated circuits” by *Brad Cox* in 1986, **Software Re-use** has been recognised as an attractive idea with a direct payoff. Building software solutions from previously developed, high-quality software components can certainly save both the costs and the time of redundant work and overall enhances solutions.

However, obtaining the holy grail of “reasonable levels of re-use” has proved elusive. A multitude of technical, process, and organisational issues have blocked and hindered progress. But despite the pursuit of co-called “silver bullets” to dramatically improve software development processes, it remains clear to those involved that systematic software re-use via component-based development is the most promising way for significant improvement.

4.2 Assembling Components

Most software development organisations have moved to object-oriented (OO) technologies because businesses believed that this would lead to significant re-use. Unfortunately, without an explicit re-use software process, most of these object efforts were unsuccessful in terms of re-use. Object-oriented technologies have remained popular for their other beneficial reasons.

The term “re-use” is something of a misnomer - no other engineering field uses this term even though the same approach is repeatedly applied – ie: the systematic design, use and re-use of standard components. This is an accepted practice in engineering fields, and we endorse this practice also for software.

An **assemble** approach can be adopted (sometimes referred to as the **buy-and-build** approach). *Brad Cox* forecast this software “revolution” back in 1986. The approach is based on taking an engineering approach to software development. Car manufacturers use pre-fabricated components, like steering wheels and seats, and assemble them using standard interfaces and procedures to build finished cars.



Similarly to building a car, building a software application is not easy. In fact, software is inherently more complex as business is much more complex and sophisticated than a device that takes you from point A to B. But by using a component-based approach to software development, it is possible to simplify the development and deployment process. By assembling, instead of buying and customizing, we avoid “re-inventing the wheel” every time.

The component-based approach provides the business benefits of “rapid application development” (RAD) for quick delivery, an enterprise-wide consistency to business rules, and caters for quick responses to changing business requirements. The component-based approach lets you mix and match best-of-breed solutions to build key software applications that are scalable, reliable, reusable, and interoperable.

4.2 Framework for Components

Software is arguably the world's most important part of the economy today. It's the catalyst and enabler, along with the required hardware technology, that has made new ways of doing business and connecting people possible.

But building powerful software solutions for business “problems” is still very difficult. Although new software development tools regularly lower the bar as to what we mean by simple, there are both market and technology-based pressures that require the building of software solutions with increasing complexity. Despite the advances in the software development tools and the methodologies, developing complex software solutions has remained and will remain labour-intensive, as ultimately, there is an intrinsic intellectual complexity in software development.

It may or may not seem obvious, but the best approach in software development is to “write less software” or rather, spend less time writing software for an equivalent level of functionality. That can show itself in several ways such as the use of languages, modelling and by using the component-based development approach. For example, a technology such as *JavaBeans* provides a mechanism to handle more complex software components and *UML* is the emergent standard for modelling. Technologies such as these put in place a framework to build or assemble software solutions from components.

The value of building software solutions from components is clear, yet we should ask ourselves why there hasn't been a flourishing supplier market of component-based solutions until now, given the advantages are so compelling? Well, it happens that the presence of the necessary technical infrastructure framework to support component-based development is only a recent advance, and has just come about in the last couple of years. For example, platforms such as *J2EE* and *.NET* platforms both support complex components. Additionally, the emergence of standard protocols such as *XML* has made it much easier for software builders to specify their components and to put them together.

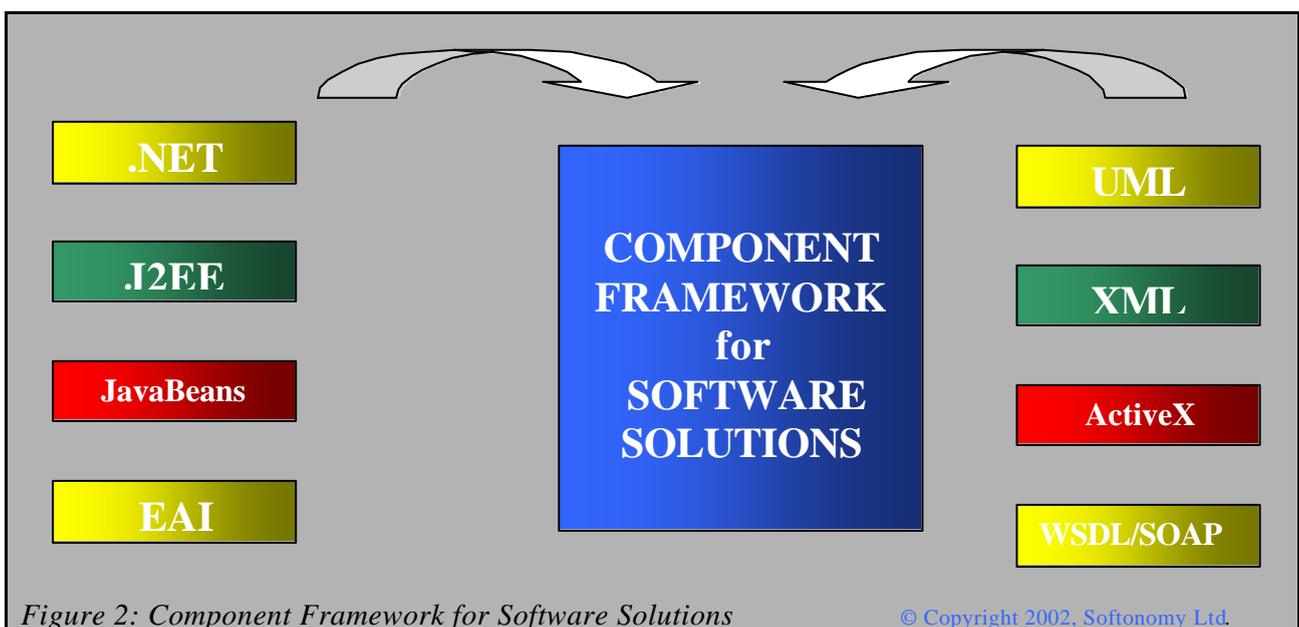


Figure 2: Component Framework for Software Solutions

© Copyright 2002, Softonomy Ltd.

5. Business Needs

While businesses in the 1980's and 1990's adopted more complex off-the-shelf packages to satisfy their ever-increasing demand for business automation, it was and is clear that not all requirements could be met by those packages. Furthermore, business dynamism has increased progressively to its current state where now businesses need to be ever more flexible and faster to adapt to change. Those that remain stationary for an appreciable length of time are losing out in terms of revenue, profits and long-term value. In tandem with this is the further maturation of the relationship that all businesses have with their *Customers*, as there is increasing Customer “awareness” and increasing Customer access to information. The Customer, more than ever, is “the King”.



Businesses know that they need *standard solutions*, but they also realise that they have unique value-adding attributes within their business that distinguishes them from their competitors and to their Customers. In fact, the larger and more complex the business, the more likely it is to be unique. For example, a Bank when compared with a Convenience Store will have much more complex requirements and can leverage technology as a differentiator to its Customers and against its Competitors.

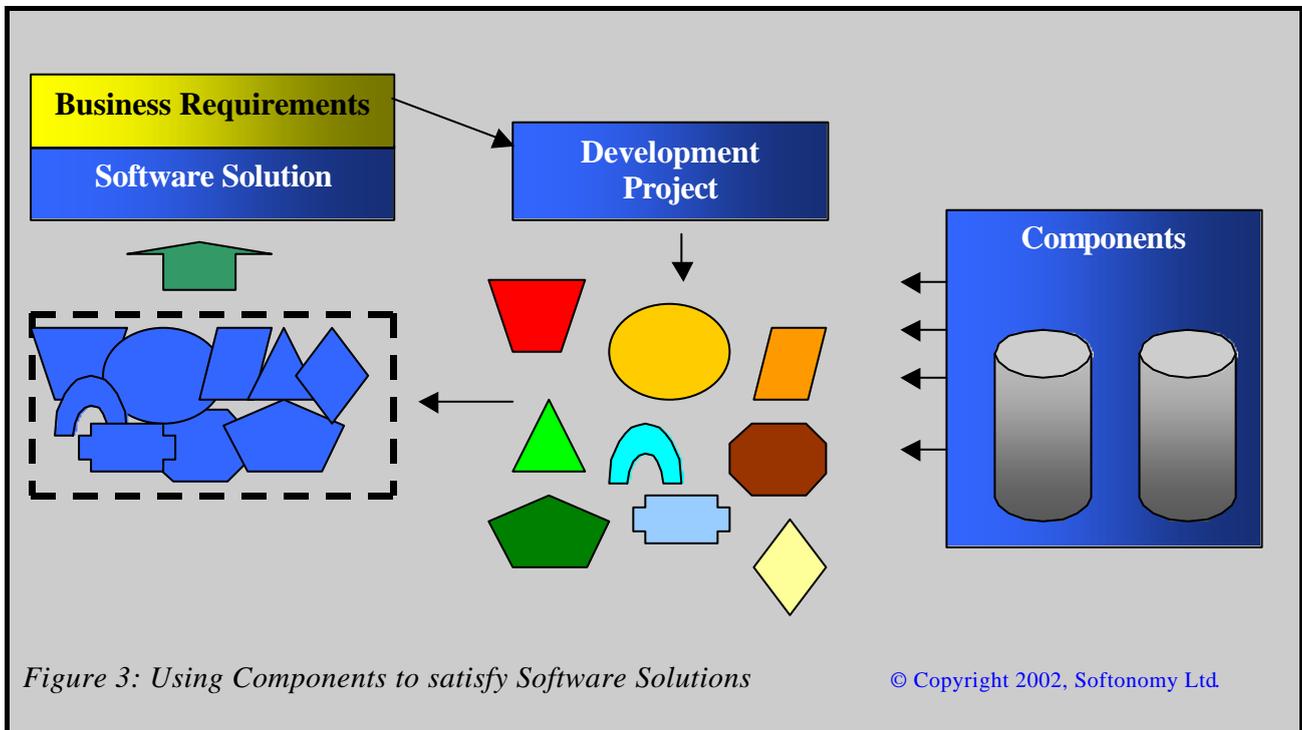
What many businesses have found when they implemented large off-the-shelf packages, is that these solutions can only go so far in providing for the information systems and automation aspects of the unique aspects in their business operations. Furthermore, the larger packages are very cumbersome when it comes to developing a solution that matches closely with the business needs. In fact, most of them were not designed from the beginning to have this level of flexibility. So, in some respects they are “the wrong tool for the job”. Using our car analogy again, the packaged software is equivalent to a Lada – ie: the same car for everyone. This would be okay if everybody had the same requirements, but people, like businesses, don’t - one size does not fit all! So, businesses are struggling with the packages they implemented as those packages are not quite delivering on their ultimate needs and in many places are very much deficient. Combined with this is the poor performance of the larger packages to meet the new and upcoming demands in a timely manner due to their unwieldiness. These factors paint a current picture of businesses dissatisfied with their information systems and software applications.

6. Tailored Solutions

Businesses have always realised that customised or tailored solutions *fit better*. However, they haven't been able to afford these due to the high development and support costs if those custom applications are built from scratch. However, it is now not necessary to build software solutions from zero by using the new component-based development paradigm. Software component re-use allows tailored software solutions to be built from robust and proven software components.

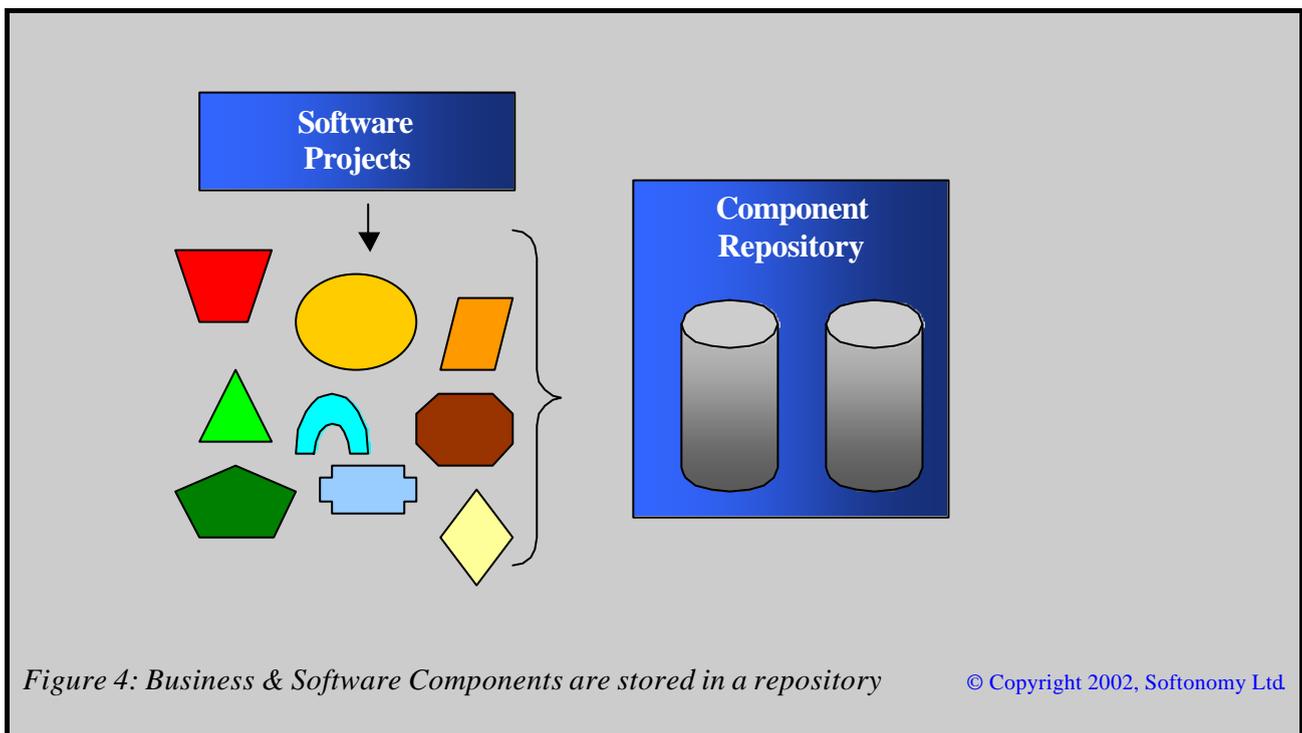
However, in many instances we have seen where Businesses state that it does not make sense to develop a custom solution, in terms of costs, etc. However, this is based on the mistaken assumption that the term “custom software” implies building from scratch. This has mistakenly tarnished the term somewhat and has made it confusing for many business decision-makers.

In fact, the advantages of custom software have not gone away, and customised software is still a *business enabler* and a competitive advantage - as long as the path of creating that software is not a deterrent in itself. It is this latter aspect that has reared its ugly head over the last decade or so. As application requirements have become more complex and more and more of business has become automated, the cost of creating custom software, each time from scratch, became increasingly less economic and hence prohibitive. So rather than custom software not being “economic”, it is the method through which it is built which must be made economic. This is where component-based software development comes in. Businesses can get their much sought after customised solutions, and because they are built using the component-based software development paradigm, the necessity to build from scratch and its associated costs are avoided.



7. What is a Component?

So far we have spoken about component-based development mainly in the context of software functional components. However, this is not the only reusable component within the software development paradigm. You can re-use all the artefacts (or deliverables), not only software, such as functional requirements, models, use cases, analysis diagrams, design documents and test cases. These and other business components, such as business processes, along with the software components combine to form what is termed a *component repository*. These components, both business and software components, can then be re-used across multiple software solutions.



But what is a software component? Well, in some sense, it's whatever you define it to be. According to the *OMG Modeling Language Specification (Rev 1.3)*, a (software) component is:

“a physical, replaceable part of a system that packages implementation and provides the realization of a set of interfaces. It represents a physical piece of a system's implementation, including software code (source, binary or executable) or equivalents, such as scripts or command files. As such, it may itself conform to and provide the realization of a set of interfaces, which represent services implemented by the elements resident within it. These services define behavior offered by instances of the component as a whole to other client component instances.”

Yes, not exactly plain english! The key to a software component is that it has to have a **defined set of interfaces** and that it actually does something, rather than describing something. We think of components as **larger grained entities** that contain a certain level of non-trivial functionality. Objects from OOP can therefore be seen as a “sub-species” of software components.

8. The Softonomy Solution

The process of “translating” business requirements into working software is a very complex undertaking, and becoming ever more so as execution architectures are extended to multiple tiers and across external networks such as the internet to Customers, Partners and Suppliers. Component-based development is now a proven method for not only increasing software developer productivity, enhancing software quality, and improving the maintainability and modifiability of the resulting software solutions, but also in terms of delivering powerful tailored software solutions giving business Clients what they need. The benefits come as a direct consequence of the component development approach: building software solutions from pre-built (and pre-tested!) components.

To provide the software solutions, **Softonomy** provides a solution building software development service working with Clients to build tailored software applications, using software and business processes components that work. These solutions not only work with the large packages already installed, but also they are customisable to closely fit a company's business.

By building software solutions using pre-built components whenever they fit, such a technique provides not only a robust and cost-efficient mechanism for software development, but also an effective framework for maintenance and ongoing support. Furthermore, with a component-based approach, adaptable Software Solutions can be built which cater for flexible enhancements, as and when our Clients future needs demand.

Softonomy have researched and developed the **Siphon[®] Intelligent Component Engine**:

- this is a highly **scalable repository** for components
- it contains **Software Components** (such as UML designs, object classes, EJB & ActiveX components, XML, etc) and **Business Components** (such as business processes)
- it is a high performing, fault-tolerant and secure engine, centrally managed. It provides a business-focused **Component Framework** environment that caters for Design Patterns, Server-Side Applications, Multi-tiered Architecture, UML Models, Software Reuse, Software Control, Software Efficiency, Business Process Management, and ongoing ROI Analysis.
- The software we build for our Clients can be applied in areas such as:
 - **Enterprise Application Integration (EAI)** (connecting to CRM, ERP & SCM)
 - The linking of “front-ends” to “back-ends” (and vice versa)
 - The implementation of **new channels** such as wireless, mobile and web/e-business
 - Solutions using the **Web Services** technologies, such as WSDL, UDDI, SOAP

Further Information

The key features of the **Softonomy Software Development Service** are:

- 1) that we build software, first and foremost
- 2) we offer a return on investment
- 3) we build tailored solutions, iteratively
- 4) we provide support, maintenance, and the path for enhancements and extensions
- 5) we build software solutions taking a business approach
- 6) we provide an intelligent component engine
- 7) we are practical about technologies
- 8) we provide skilled software development resources
- 9) we have experienced software developers with sector knowledge
- 10) we deliver effective project management

We focus on software development to bring our Clients cost effectiveness and efficiency, technology flexibility, effective software solutions and to provide a future solution path.

For further information on **Softonomy** and our other white papers, please refer to our website located at www.softonomy.com

or send an email with any questions you may have to: info@softonomy.com

Alternatively, contact us as follows:

Tel: +353 (0)86 821 0917

Fax: +353 (0)1 284 6381

We look forward to hearing from you.
