Developing Composite Applications

Comparing the Total Cost of Ownership

Written by: Tim Jennings and Rob Hailstone

Published December 2006

▶ MANAGEMENT SUMMARY

The recent emphasis for IT departments on systems consolidation and compliance with information legislation, has led to a backlog of application development work which must be accomplished in an effective and timely manner. Addressing this need, significant benefits in both speed and development costs can be realised by deploying service-oriented composite applications, which draw together business logic and data sources from multiple underlying systems.

There are multiple benefits that derive from this service-oriented approach to development, which add up to substantial cost savings for the organisation: a simplified development environment reduces the amount of coding required, and allows existing development staff to be deployed without the need for extensive retraining; there is an opportunity for extensive service reuse, and consequent savings in development time; rationalisation of legacy processes helps to reduce the total number of processes that must be managed; and by using a standard set of simple protocols, the effort of application maintenance is considerably reduced.

The environment for building composite applications requires several components, but organisations often end up using multiple point solutions to meet these requirements, rather than an integrated toolset. This leads to several problems: there is no common management layer, so the applications can themselves become silos; there is no central repository of services, which leads to low reuse and duplication; and disparate toolsets are used in different parts of the organisation, resulting in services that are not fully compatible from both technical and business perspectives. Butler Group therefore considers that a well-integrated suite of components is a prerequisite for the successful development of composite applications. This reduces the timescales and costs of development, makes applications easier to manage and maintain, and reduces the risk of fragmented applications and processes. Importantly, it also provides the foundation for a unified, enterprise-wide approach to a service-oriented architecture, maximising the speed and breadth of the overall adoption curve.

To provide evidence to support this assertion, research was undertaken based on interviews with customers involved in composite application development. The study examined the costs of a typical project developed using Sun's integrated environment, which since re-branding has become the Sun Java Composite Application Platform Suite (CAPS). The costs were compared to a traditional approach using separate best-of-breed tools in a JAVA EE environment. The study demonstrated benefits across all phases of the application lifecycle – design, development, testing and deployment, and subsequent maintenance, and concluded that an integrated suite offers a potential 50% reduction in the initial project development phase (design to deploy) compared to a traditional development approach, and up to a 58% saving in Total Cost of Ownership (TCO) over a three-year period, when ongoing maintenance costs are taken into account. Applying average developer cost rates to this example would give a total cost saving over three years of €180,000 (approximately US$223,000) on a single project.

Since the original interviews were conducted, Sun has extended the scope of its SOA offering by creating the Sun Java Composite Application Platform Suite (Java CAPS). This is an integrated toolset for business process integration and the building, execution and management of composite applications. It provides a comprehensive framework for service-oriented application development, using open standards. CAPS brings together the capabilities of the SeeBeyond ICAN product (the result of the acquisition of SeeBeyond, completed in 2005), Sun's Application Server 8.1 (the reference JAVA EE application Server), plus Sun's Directory, Portal, Single Sign-on, and Sun Enterprise Development Tools. The suite includes comprehensive application connectivity, message management and transformation, security, bulk data movement, identity management, and the necessary run-time infrastructure components to provide a fully-functional environment. Most importantly, from the viewpoint of this paper, it includes the studio capabilities to design and create entire composite applications, from the graphical user interface through to the orchestration of the application components. It also provides the activity monitoring to "close the loop" – providing information on the actual usage and service levels experienced to enable the ongoing optimisation of processes.

Butler Group believes that Sun Java CAPS delivers a powerful proposition for improving the time-to-value for integration projects as well as delivering rich functionality through composite applications, and which can significantly reduce the TCO when compared to the use of separate point solutions. For organisations that wish to take a still-broader view of their software infrastructure, Java CAPS itself is one of a number of suites within Sun's Java Enterprise System and Solaris Enterprise System software bundles, each of which offers full integration across the whole spectrum of products.

The range of licensing options provided by Sun (per employee subscription or per annum) complements the functionality of the Java CAPS suite to make it an attractive option for organisations wishing to deliver composite applications in a cost-effective manner. It will also help organisations to be more flexible in their software infrastructure investment planning. Sun's open source distribution makes it simple to download the products and construct working applications before electing to license the supported products.

▶ COMPOSITE APPLICATION CHALLENGES

Most IT departments are now in the process of transitioning from a post dot-com crash "bunker" scenario to one of preparation for modest growth in a changing market. Since 2001 the emphasis for the majority of IT departments has been on cost reduction through infrastructure consolidation and software rationalisation, together with modest investment to meet the needs of compliance with information legislation (such as Sarbanes Oxley, the Basel II Accord in the banking industry, and Freedom of Information legislation in the public sector). There has also been an increased focus on improving IT governance, including IT-to-business alignment, and the need for improved metrics on IT value delivery.

This focus has in many cases generated a backlog of application development work, both from regular line-of-business requirements, and from senior management keen to undertake broader business transformation initiatives, building on the benefits of an updated infrastructure. It is evident that having realised savings through consolidation and leaner operational costs, there is even greater pressure to deliver applications that are well-aligned with the business, in a cost effective and timely manner. This has created the present environment of readiness to embark on strictly-controlled investment to enable the delivery of automated processes and functionally-rich applications, targeted at increasing the organsation's responsiveness to market pressures without having to resort to a major "rip-and-replace" of established systems.

Whilst consolidation has been beneficial, it has occurred mainly within a particular architecture, for example, reducing the number of UNIX servers, or pooling storage capacity, and has done less to address the problem of multiple architectures that exists for many organisations. Heterogeneity has become a long-term feature of the IT landscape, which we must learn to exploit. This has led to greater awareness of the significant benefits in both speed and flexibility that can be achieved by using service-oriented composite applications, resulting in substantial cost savings.

A composite application can be defined as a programmatic solution that addresses a specific business problem by drawing together business logic and data sources from multiple underlying systems. Typically a composite application will be associated with a business process, and may draw together several process steps, presenting them to the user through a single interface that is customised to the requirements of the business need.

Despite the consolidation efforts that have taken place, in most organisations the majority of business processes still span multiple operational systems. A composite application is designed to overcome this issue because it is process-centric or task-centric, rather than being system-centric or application-centric. Because the business logic is no longer embedded within a particular system or systems, it provides greater flexibility when an organisation needs either to change its business processes, or to update the underlying systems: this is one aspect of what is often described as a 'loosely-coupled' architecture.

In many respects, this move towards composite applications represents the next step of maturity in application development, combining many of the benefits of bespoke development (a customised solution that tackles a specific problem), with those of packaged applications (a common foundation that requires minimal coding and is easy to maintain). This reflects the evolution that occurs in many IT domains, where technology moves from a do-it-yourself methodology, to a largely pre-fabricated modular approach which still allows flexibility for individual organisations.

If businesses are to reap the benefits of composite applications, then the development environment selected must provide a highly-productive framework for creating applications. It is also vital that this framework is built on open standards, so that these applications are independent of the underlying technologies and architectures; that they can be deployed into any technology environment; and that organisations can find the necessary skills within their existing development staff, or can acquire them at advantageous market rates.

Whilst composite applications offer undoubted benefits, they are not per se an automatic solution to the application development backlog, but must be coupled with an effective methodology and an effective toolset. As with any application development concept, a disconnected approach will lead to increased development times, fragmented business logic, and high-maintenance overheads.

▶ BENEFITS OF A SERVICE-ORIENTED APPROACH

One of the corollaries of separating business processes and business rules from the core underlying systems in a loosely-coupled style is to encapsulate and expose the functionality of these systems in a modular fashion. This service-oriented approach to composite application development involves designing and creating reusable services from existing systems, and combining these services according to the definition of a specific business process – a technique often described as ‘orchestration’.

The key benefit to this approach lies in the fact that many business functions occur in multiple processes, offering an opportunity for extensive reuse and consequent savings in development time. Examples of such services include reading or writing a customer record; looking up an account balance; calculating a tax value or shipping cost; and initiating a work order. Responding to and encouraging this trend, both software developers and packaged application vendors are now routinely designing their products to expose business logic as a set of services.

For a small organisation or single business unit, the core set of business services is reasonably intuitive, but beyond this limited scope it becomes necessary to introduce a formal way of cataloguing the services that have been created, so that the degree of reuse is maximised across the whole organisation. It is also important that services are designed to be self-contained in nature, so that they fit as wide a range of applications as possible; that they are built using a standard set of simple protocols; and that a service is represented by a single version that is easy to maintain.

For the enterprise, the skill set required to construct a composite application from a set of services is considerably less than that needed to write a similar application from scratch, and involves less coding. This makes it easier to redeploy existing developers, without having to undergo lengthy retraining in technologies such as JAVA EE, which often takes between 9-12 months. Because these applications are process-driven, it also supports better business alignment by providing a common point of reference for both business analysts and developers.

Taking a process and service-based approach to application development also helps an organisation to rationalise its processes. Siloed applications often include process fragments that are locked away, difficult to change, and are duplicated across several systems. The ability to define and reuse sub-processes as services is a powerful technique to help reduce the overall number of processes that must be managed.

▶ THE CASE FOR AN INTEGRATED TOOLSET

Butler Group believes that the advantages of the service-oriented, composite application approach are clear, but to achieve these benefits also requires a change in thinking on the tools that are employed. The process of building an SOA with associated composite applications requires, as a minimum, the following components: a development tool for defining and creating services, a graphical process definition environment, a tool for generating a user interface, and an execution ‘engine’.

In many cases, however, organisations end up using multiple point solutions, either because there is a lack of overall programme and project co-ordination, or because they are frustrated at the lack of functionality within individual tools. Each element is selected from a different toolset, often from different vendors, for example, combining an integration package with a modelling tool, a development environment, and a business process management engine. Because there is no integration between these elements, and they lack a common management layer, the resulting applications themselves become silos.

Without a central repository of services, particularly in a large organisation, users embarking on a project have no easy way to ascertain which services already exist. Consequently, there is a low level of reuse and current services are duplicated, leading in turn to further confusion (and a magnification in maintenance challenges). It is also not uncommon to find that different toolsets are in use within different parts of an organisation, resulting in services that overlap in both business functionality and technical standards, but which are not fully compatible from either perspective.

It is also important that there is a common understanding and approach to the organisation's SOA design, and we firmly believe that this should be process-led, i.e. top-down, rather than being built from the bottom-up. Without this common approach, that is fostered by the use of an integrated toolset, business analysts and developers are not encouraged to approach a project from a process perspective, and there is a poor understanding of the services that will be required to create an application.

Further problems can arise from a lack of flexibility in the creation of an interface for a composite application: in many platforms, the application is exposed solely via a portal front-end, which tends to provide a good data-centric view, but is not as strong in the management of processes. The lack of an integrated management layer also means that it is difficult to manage the application without writing specific management functionality into each service or application, whereas in an integrated suite, this functionality can be generated automatically.

Butler Group therefore considers that a well-integrated suite of components is a prerequisite for the successful development of composite applications. This reduces the timescales and costs of development, makes applications easier to manage and maintain, and reduces the risk of fragmented applications and processes. Importantly, it also provides the foundation for a unified, enterprise-wide approach to a service-oriented architecture, maximising the speed and breadth of the overall adoption curve.

▶ COMPOSITE APPLICATION DESIGN PRINCIPLES

The key components required for the effective development of composite applications can be summarised as follows:

- A centralised repository to simplify the discovery of existing services and maximise their reuse.
- The simple definition of integration points into existing systems, built on basic Web services protocols, including UDDI, SOAP, and WSDL.
- The use of automated data mapping and transformation services for data level integration.
- A graphical environment for the definition of business process flows, including control structures such as conditional branching, and parallel processes.
- The ability to exchange process definitions with other systems and with business partners, in a standard format.
- User interfaces that can be easily generated from existing services, and customised as required.
- An application development environment that minimises the amount of coding required, and reuses existing developer skills.
- A framework based on open standards that can be deployed across a heterogeneous environment i.e. that is available on multiple platforms.
- Robust master data management facilities, enabling a single view of key business entities such as customers or products, over multiple underlying data sources.
- The ability to automate business processes, reducing the need for manual intervention.

▶ TCO MODEL FOR COMPOSITE APPLICATIONS

The benefits of an integrated suite have a dramatic impact on the Total Cost of Ownership (TCO) of a composite application development framework, compared to a traditional development approach using separate tools. In order to quantify this benefit, Butler Group and SeeBeyond have undertaken a study based on data gathered in recent customer interviews. The study looks at the dimensions of cost, time, skills, and risk across the four phases of the development process – design, build, test, and manage.

Factor \ Phase	Design	Build and Develop	Test and Deploy	Manage and Maintain
Development Time	✓✓	✓✓✓	✓✓	✓✓✓
Skills Acquisition and Retraining	✗	✓✓✓	Neutral	✓
Degree of Project Risk	✓	✓✓	✓	✓✓
Overall Project Costs	✓	✓✓✓	✓✓	✓✓

Table 1: Summary of Impact on Development Phases

The matrix in Table 1 shows the relative impact in each of these facets of a project, on a scale from highly negative (×××) through to highly positive (✓✓✓):

The study used the information gathered from customer interviews, and applied this to the detail of a typical project. Table 2 shows the costs of developing a composite application for a single business process, with an integrated suite such as Sun's Java CAPS, compared to a traditional approach using separate best-of-breed tools in a JAVA EE environment. The figures shown are the number of weeks of development time taken for the project in each phase.

Area	Integrated Suite (weeks)	Traditional Development (weeks)	Saving
Design Phase – requirements capture – specification – proof of concept – process modelling – data modelling	9.0	12.0	25%
Solution Build and Development – integration to existing systems – business logic development – GUI development – management functions	1.0	7.0	86%
Testing – integration testing – system testing – user acceptance testing – application deployment	3.5	7.0	50%
Deployment – application deployment – running validation	2.5	4.0	37%
Project Management	@15% 2.4	@20% 6.0	60%
Project Contingency	@20% 3.2	@25% 7.5	57%
Total Design-to-Deploy Time	21.6	43.5	50%
Maintenance Years 2 and 3 – application maintenance	8.6	29.0	70%
Total Savings	30.2	72.5	58%

Table 2: Detailed Comparison of Project Development Costs

Key Conclusions from the TCO Study:

- Use of an integrated suite delivers a significant reduction in project time, particularly during the development phase, since very little coding is required.
- Business analysts will require training on the process design tools, which may increase the design time for an initial project, but this is more than offset by the fact that existing developers (from practically any development background) can be deployed to both initial and subsequent projects, without the need for retraining in complex skills such as JAVA EE.
- The reduction in coding and in overall solution complexity leads to a general reduction in testing time.
- Deployment effort is significantly reduced, due to close integration between development and deployment environments.
- There is a percentage decrease in the project management overhead rate, due to the use of a common platform and the reduction in handover effort between successive phases of the project.
- Project risk is reduced over all phases, with a consequent reduction in the contingency rate that is applied to the project. This is due to factors including improved communication between business and IT functions, reduced coding, built-in management functionality, and the ability to work with existing skill sets.
- There is a substantial reduction in maintenance costs for the project in succeeding years, due partly to the reduction in code, but chiefly due to the added flexibility that comes from a process-driven approach to development. When business requirements change or core applications are replaced, the loosely-coupled relationship between process and underlying systems removes much of the complexity.
- The overall conclusions of this study were that an integrated suite offers a potential 50% reduction in the initial project development phase (design to deploy) compared to a traditional development approach, and up to a 58% saving in total cost of ownership over a three year period, when ongoing maintenance costs are taken into account.
- Applying average developer cost rates to this example would give a total cost saving over three years of €180,000 (approximately US$223,000).
- A service-oriented development approach and an integrated suite can provide further ongoing benefits to an organisation, as the effect of service reuse leads to additional cost and time reductions in successive projects.

▶ SUN'S APPROACH TO COMPOSITE APPLICATION DEVELOPMENT

Sun's Java Composite Application Platform Suite (Java CAPS) is an integrated toolset for business process integration and the building of composite applications. It provides a comprehensive framework for service-oriented application development, and makes use of open standards at each level of the model, with the philosophy that layers should be interchangeable with any other elements of an organisation's infrastructure.

The functional architecture of Java CAPS is clear and consistent: at the bottom of the stack are the enterprise applications, data sources, and links to trading partners that comprise the organisation's ecosystem. Using the Java CAPS extensive range of adapters, business functionality from these systems is exposed as a series of granular Web services. At the next level, these services can be orchestrated into business process flows that address specific operational requirements. Finally, these processes are combined with a graphical user interface, consisting of data entry and presentation functionality to create powerful composite applications.

The foundation of the Java CAPS is the eGate Integrator platform, which is based on Sun's Java System Application Server (or other certified JAVA EE application server), and which uses both standard JAVA EE Connector Architecture (JCA), as well as Sun's own range of adapters. This platform is highly scalable and provides guaranteed exactly once delivery, which is critical in many mission critical environments. It has a single open repository that stores all metadata for the system, including processes, objects, services, applications, and transformations and exposes all integration points as Web services, with Java or XSLT used for transformations.

In addition to the Enterprise Designer for building services from existing assets, Java CAPS includes the Java Studio Enterprise and Java Studio Creator products. These provide respectively the workspace for creating new services and for creating a rich user interface. Together they provide a common integrated development environment across the whole suite. The IDE allows business and technical users to share a consistent view allowing connectivity to be designed using a graphical editor, or a built in Java (or XSLT) editor. It is the use of this common interface to the whole Java CAPS that enables an intuitive approach to designing end-to-end processes, across all levels, from connectivity and orchestration, right through to presentation. Much of the work is visual in nature, with components being dragged and dropped onto a working canvas. This minimises the coding required, and enables existing developers to be quickly productive without extensive retraining.

The orchestration of business processes is carried out by Sun's eInsight Business Process Manager. This generates process orchestrations in the form of Business Process Execution Language (BPEL), which has become the accepted standard for the interchange of business process definitions.

Master data management capabilities are provided by Sun's eView Studio which allows the creation of applications to provide a single view of an object that is not otherwise defined within enterprise systems. Wizard-driven, and using fuzzy logic, eView creates a cross-index for entities such as individuals, businesses, or products, that are referenced across multiple application and data silos.

Sun's eBAM Studio module adds process analysis capabilities, described as Business Activity Monitoring (BAM), which combines the use of business metrics with business process flows. It includes an editor for defining key performance indicators, generating alerts according to user-defined criteria, and presenting these through a dashboard format.

Java CAPS also supports integration outside the organisation, with B2B links to trading partners through its eXchange Integrator module. This has support for many on-ramps and emarkets, and enables the organisation to create packaged functionality, including composite applications, for easy download and configuration by its partners.

To support diverse data requirements, Java CAPS includes powerful ETL capabilities for bulk data migration and populating data warehouses, Butler Group believes that Sun Java CAPS delivers a powerful proposition for a new breed of composite applications, which can improve the time-to-delivery for integration projects, and which can significantly reduce the TCO when compared to the use of separate point solutions.

With the addition of Java CAPS to the Java Enterprise System, Sun is now able to offer a complete solution to support SOA and the creation of composite applications. It combines core integration capabilities for both Java and legacy environments, business process modelling and orchestration, convenient interface design, metadata management, and strong development tools into a comprehensive platform that is built on open standards.

In combination, the two products have the potential to offer an even stronger TCO advantage: because of the integration between Java ES and Java CAPS, developers will have to spend less of their time coding the integration between systems components such as:

- Portal Server (JSR 168 portlets, WSRP).
- Web Server (web presentation).
- Directory Server (LDAP authentication and authorisation).
- Access Manager (access controls).
- Identity Server (identity provisioning).

By integrating Java CAPS with the platform technologies in Java ES, developers will be able to develop more rapidly the presentation layers of their applications and will benefit from linking both people and services to the identity and access management layer. Both are critical to successful SOA application development and the requirements of a SOA architecture.

Most importantly, Java CAPS is able to deliver rapid business value, by supporting an incremental approach to projects, and the ability to evolve existing IT assets into a modern SOA, as opposed to a rip-and-replace approach. The breadth of solution offered by Sun, together with the impact of recent acquisitions made by the company, has also strengthened Sun's Systems Integrator (SI) partner network, and the company is well placed now to help its customers realise increased benefits from their existing IT investments through the use of composite applications.

Contact Details

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara
CA, 91016
USA

Tel: +1 650 960 1300
+1 800 555 9SUN

E-mail: sun_seebeyond@sun.com

www.sun.com

Headquarters:

Europa House,
184 Ferensway,
Hull, East Yorkshire,
HU1 3UT, UK

Tel: +44 (0)1482 586149
Fax: +44 (0)1482 323577

Australian Sales Office:

Butler Direct Pty Ltd.,
Level 46, Citigroup Building,
2 Park Street, Sydney,
NSW, 2000, Australia

Tel: + 61 (02) 8705 6960
Fax: + 61 (02) 8705 6961

End-user Sales Office (USA):

Butler Group,
245 Fifth Avenue, 4th Floor,
New York, NY 10016,
USA

Tel: +1 212 652 5302
Fax: +1 212 202 4684