

The Missing Link of SOA

Organisations adopting an SOA approach are missing out if they ignore the potential for re-use of User Interfaces argues **David Davies** of Corizon.

Agile Development is a concept rather than a formal methodology. It accepts that requirements can evolve during a project's lifetime and acknowledges that this is unavoidable. It suggests that project processes should be set up to welcome and accommodate this change, rather than trying to restrict or avoid the inevitable.

Service Oriented Architectures (SOAs) are gaining momentum as an approach to organising and building enterprise IT infrastructures and rightly so. SOA addresses the twin key needs for lower IT costs and greater flexibility. Software assets from many types of components can now be easily exploited in new, sophisticated solutions, without complex integration projects.

Current SOA thinking focuses on the lower levels of an enterprise IT infrastructure - how to create, manage and combine business services that provide data and logic. However, its core principles of re-usable well-defined services and loose coupling aren't applied at the User Interface (UI) level. UI is treated as a throwaway item and must be redeveloped where it is needed. The result is that composite solutions for users are complex and expensive to build and maintain.

Not going all the way

In an SOA environment, the integration and re-use of data and logic is simplified by adopting two principles. Firstly, business services with defined interfaces are created to allow functionality to be built once and consumed as required. Services package up new and legacy functionality, making it possible to exploit it more easily and cutting the need for stovepipes as assets are built once and re-used as needed.

Secondly, the provision of the services is separated from their consumption - so that the building blocks don't need to "know", for example, about how they will be recombined and orchestrated. This is the only way for re-use to be scalable and efficient. With complex implementation code hidden behind the business service interface, the orchestration process becomes comparatively rapid and lightweight to develop.

Unfortunately current SOA does not go all the way to the user - it does not treat UI as a re-usable component in the same way as business logic. Today, the only option available to create a UI for a group of users in an SOA is to turn to bespoke development that leverages the business services. This leads to stovepipes being created for each requirement and user group, with functionality being developed and maintained many times over.

Agile development is a concept rather than a formal methodology.

Alternatively, all possible permutations required by the user groups have to be catered for in a single complex UI development. In either case, the solution is locked in an escalating amount of code, which makes it inflexible, slow to build and unresponsive to business and user group need. The next result is that the benefits sought from the SOA can be undermined.

A related problem for SOA adopters not addressed by the current data centric approach is what to do with valuable application UIs - bespoke or packaged, internal or partner provided - when they are needed the context of an integrated solution. The implicit assumption - that the UI should be thrown away and the functionality recreated in the SOA - has not been widely challenged but presents a major barrier to delivering business requirements.

Enter: the "service-enabled" UI

The problems with composite application UI development: proliferation of stovepipes, incorporation of legacy functions and tight coupling of UI development to its consumption by different user groups echo the issues that SOA was introduced to solve at a data and business service level. They should also be addressed by extending SOA principles to the UI.

Two steps are needed: firstly to introduce the idea of "service-enabled" UI in addition to the more commonly understood data-oriented services. These constitute building block components of UI functionality, which means that presentation, validation and low-level user tasks can be embodied in code once and provided for re-use as running services. A second requirement is to be able to easily combine and tailor this service-enabled UI into a seamless solution for each user group, with no involvement of the service provider. This requires "UI orchestration", with specialised user-centric capabilities distinct from those provided by conventional BPEL-style data service orchestration.

Until recently, this functionality has not existed. New software can now enable this. It allows, the presentation layer of existing applications - browser or Windows based - to be broken down into service enabled components and then rearranged, along with UI functionality from other applications and new UI that leverages the business services. The end result is a new composite UI that more closely matches the everyday business needs of end users.

As a result we can now extend the SOA concept to include a new class of re-usable assets - the extremely valuable application UI. Integration projects are cheaper and easier, due to the simplicity, in development and maintenance terms, of pulling together existing UI functions, compared to the alternative of creating them again each time they are needed. The promise of flexibility bears fruit, too: different user groups can be provided with highly functional optimised UIs. ■■

Corizon is the leading provider of User Process Management (UPM) software that simplifies and customises end user experiences of IT applications. The company's platform allows businesses to define, implement and monitor user activities across application boundaries.

www.corizon.com/