SaaS CRM application. It displays in a browser, is easy to use and works well. But to process an inbound customer call, an agent must do an account lookup in a legacy mainframe application (requiring a time-consuming, separate login), which displays in a separate window. Information in the mainframe window must be copied and pasted into the SaaS CRM application to continue. Data from the CRM application then must be copied and pasted into the Notes window to keep a journal of the call. The agent must keep track of compliance-critical lists of information from a printed notebook, with no on-screen prompts or records.

For this call centre, moving to a remotely-hosted CRM application may have solved some budget and infrastructure problems, but the need to switch between multiple applications and tedious manual processes continues to degrade productivity, and clearly compliance and data integrity could be an issue.

## Why traditional integration won't work

Traditional integration approaches rely on either available APIs or connectors, or access to applications' source code for coding purposes. In some cases, undertaking traditional integration projects will be effective. In many cases, however, it will not. There are several reasons why SaaS presents some very real challenges to traditional integration techniques:

- Lack of, or minimal depth of APIs. Integration with SaaS applications is limited to functionality or data exposed by the SaaS vendor's developers. You no longer own or even have access to the underlying source code.
- Reliance on SaaS vendors. Any non-standard integration projects will depend on your vendor's development schedule, not your own.
- Core applications change. Enterprise desktop applications are upgraded and previous SaaS integration APIs or code are invalidated.

It was once thought that service orientated architecture's (SOA) adaptability and custom focus would solve problems like this. But changes to the architecture can ripple across different areas and require costly and time-consuming work, such as recoding previous solutions.

So presentation-layer integration, which comprehensively integrates all applications on the user desktop, remains a key challenge, and an important one for productivity. Virtualised and SaaS applications can be delivered to the desktop, but they are still used in Windows-native interfaces. If those interfaces don't function together, then the human operator is the integration tool. If your key business objectives include things like better customer service, a 360°customer view, better up-sell and cross-sell or other goals that can be measured in terms of productivity or capacity, then putting the integration burden on the user could be a less than optimal solution. Then add the subjective factors like training new users in complicated, labour-intensive tasks.

## Second integration approach

The good news is that there are other integration approaches that don't require APIs or access to source code for development purposes. One approach in particular is ideal for the world of cloud computing.

Presentation-layer integration is quickly gaining popularity because it tackles the integration challenges of the cloud head-on. The one common denominator across all

To paraphrase the report, from now until 2013, SEAP technology will be "opportunistic and architecturally simple application development among Global 2000 enterprises." Meaning, SaaS won't replace most sets of desktop applications anytime soon, presenting the integration dilemma.