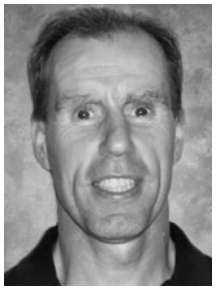
Five factors to consider before embarking on custom application development

by Patrick Dixon



Patrick Dixon
Manager, Deloitte Consulting

Patrick Dixon is a manager with Deloitte specializing in SAP Enterprise Portal and SAP interface design and implementation. He has over five years of experience Web-enabling and integrating SAP systems with SAP Enterprise Portal, SAP Internet Transaction Server, SAP Exchange Infrastructure, and SAP NetWeaver. Patrick was a key speaker on portal implementation and content integration at SAP NetWeaver/BW and Portals 2005 and other portal events. Prior to moving to the US in 1996, he was a freelance consultant in Great Britain. Patrick graduated from Bath University in 1983 with a master's degree in computer simulation and subsequently obtained an MBA from City University, London. You can reach him at padixon@deloitte.com.

Custom code, even just a little, can have a dramatic, beneficial result. For example, removing display-only fields that data-entry users don't need can simplify an order form and speed up the data-entry process. Too much custom development, however, can leave you with a cumbersome system, too complicated to manage. Adding too many fields to an order form, for example, so all possible options appear on one screen, can overwhelm users and slow down the data-entry process.

Custom code is also expensive. It must be administered, maintained, enhanced, upgraded, tracked, and eventually migrated or retired. If your custom development is spread across multiple systems and languages and has different dependencies (e.g., platform settings, configuration, and so on), the situation is exacerbated. Customized software is not supported by SAP when you are upgrading, and when problems arise at any point, they can be difficult to resolve, especially on your own. Any way you look at it, the best rule to follow is to turn to custom development only as a last resort.

But how do you determine whether custom development is unequivocally required for your company to meet its goals? If it is and your company is running on SAP NetWeaver, how do you choose the best of a range of custom development options?

This article presents five factors that you need to consider before you embark on customization. It includes suggestions for investigating whether or not customization is necessary at all, and presents several customization alternatives that will enable you to meet certain needs without the expense of custom code. If you find that customization is necessary, you will need to weigh the advantages and disadvantages of available development tools, and evaluate your in-house skill sets so you can anticipate the challenges that lay ahead. To help you do just that, this article also looks at some of the available customization options, with a focus on Web presentation, which is emerging as the preferred presentation for SAP content. Finally, this article offers my opinion on the future direction of SAP development, which you should consider to ensure that your coding efforts do not quickly become obsolete.

Three solid arguments for customization

1 **The out-of-the-box solution you have chosen is too complicated for your target audience**. Over-complicated user interfaces waste time and effort, which translates into business dollars lost. To construct a less-complex interface, use the Visual Composer tool. It allows you to use existing BAPIs and RFCs, but if the interface parameters of the RFCs or BAPIs change with a future version of SAP, it is a fairly simple task to reconstruct and redeploy the iView using the Visual Composer workbench.

2 **Custom modifications are required in order to realize maximum SAP functionality**. An example of this is *eventing* within SAP Enterprise Portal. Eventing is the passing of data between iViews, and it lets users click on one iView while simultaneously displaying related data in other iViews on the same portal page. I find that users love eventing, but the programs behind the iViews have to be coded to make use of the eventing utility, which is provided by the portal in the form of JavaScript.*

3 **You need to undertake custom coding when you want to use built-in development tools**. An example is the proprietary HTML Business for Java (HTMLB) functionality, which is a Business Server Pages (BSP) extension that SAP provides to save development effort when coding objects such as tables in Java programs. With HTMLB, the coding of the table is considerably simplified because HTMLB takes care of the paging and display of data in the table.

* One click changes the data in all evented iViews.

Five factors to consider before customizing

Each organization has its own special needs and requirements that may require customization (see the sidebar above, which outlines three scenarios that merit customization). When determining whether customization is necessary to achieve your goals, weigh the following factors:

- Know when customization is really necessary.
- Consider alternative customization options to coding.
- Understand both the advantages and disadvantages of a particular tool set.
- Match the available customization tools to your in-house skill set.
- Understand the future direction of SAP development.

In the following sections, we take a closer look at each factor in turn to provide you with a comprehensive guide to making the right decisions for your own organization.

Know when customization is really necessary

It may seem obvious to state that custom code should be considered only as a last resort, but time and again, I find people diving into custom coding before they have examined all their options. Some options may not remove the need for custom coding entirely, but they certainly can reduce the amount of custom coding required.

My advice is that if you can get 80% of your desired functionality without custom coding, it is probably not worth the effort to resort to custom coding to get the other 20%. Remember, custom coding is an expensive option in terms of time, money, and resources. The

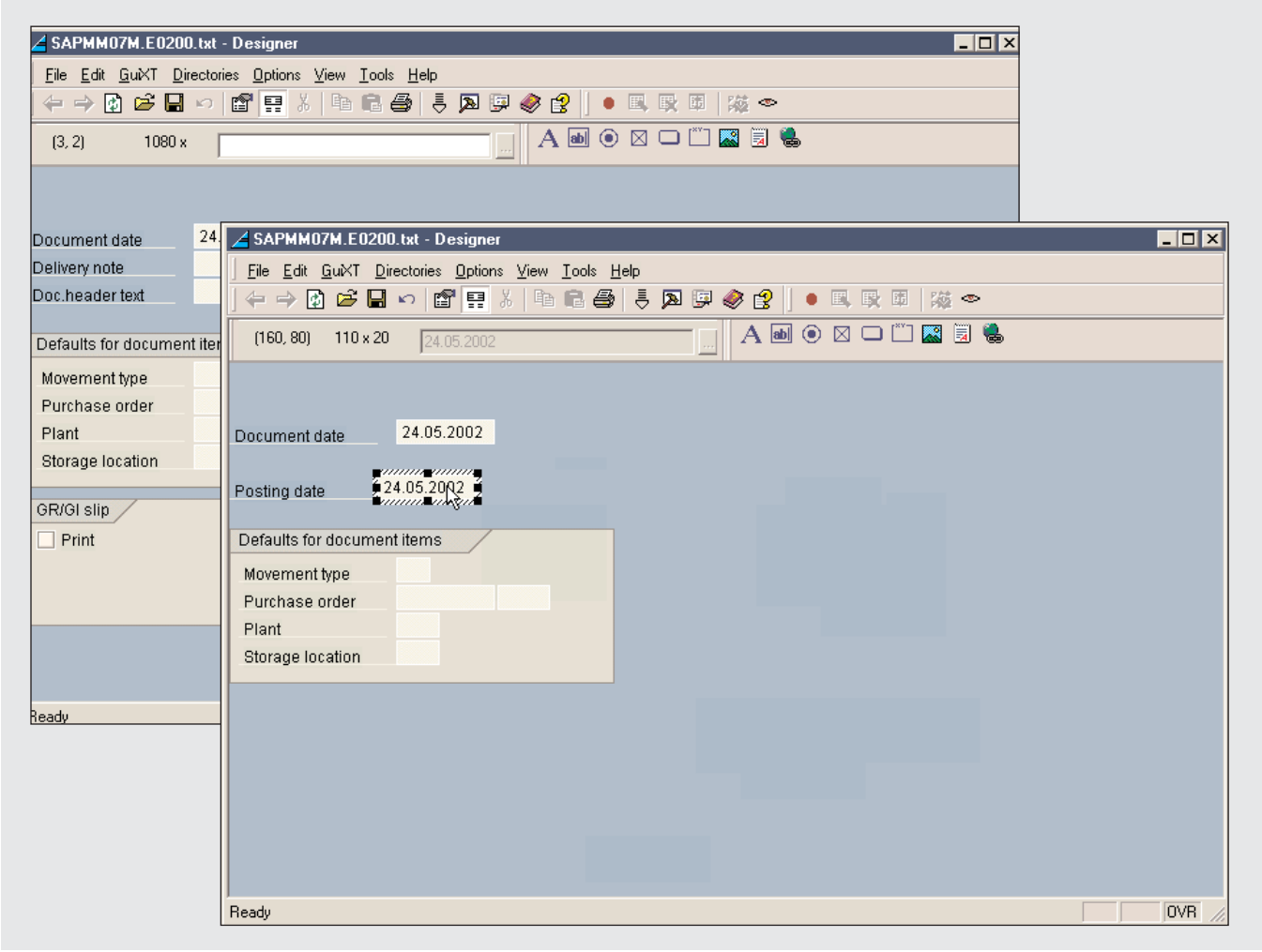


Figure 1 Standard SAP R/3 screen modified with Synactive's GuiXT tool

effort directed at the remaining 20% may be far better used on another project that ultimately could deliver a superior ROI.

Custom coding should therefore be used only when all available out-of-the-box options have been explored and determined unable to provide the functionality to the user that is critical for the project to succeed. I had a client who wanted a production-line worker to be able to order parts; however, the complexity of the out-of-the-box transaction often led to costly mistakes in data entry. Here, a simplified custom front end for the transaction that reduced data entry to a handful of key fields with selection options and no tabs was necessary to reduce data-entry errors.

Consider alternative customization options to coding

The advantage of coding a custom interface is that you have a significant amount of control over what is finally produced; however, I consider coding as my customization option of last resort because manual typing — at least my typing, anyway — inevitably leads to errors. Several visual tools can give you the custom interface you want without the need for typing:

- GuiXT, the oldest of these visual tools, is a third-party product from Synactive that allows streamlining (simplification) of SAP transactions (see **Figure 1)**. This is a great option to use if

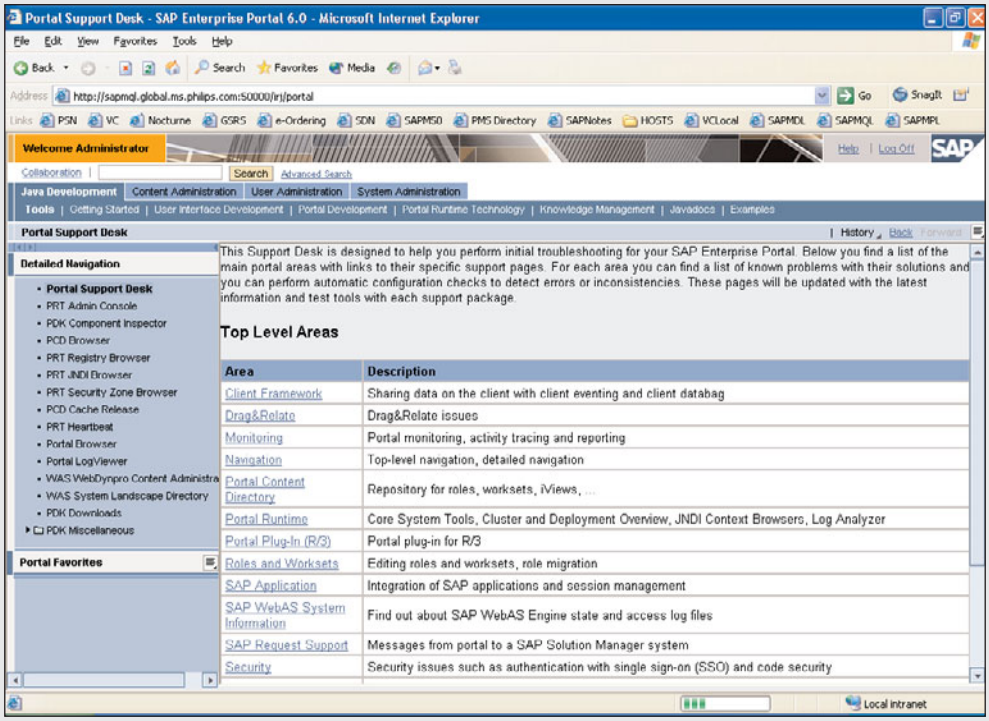


Figure 2 Sample business package

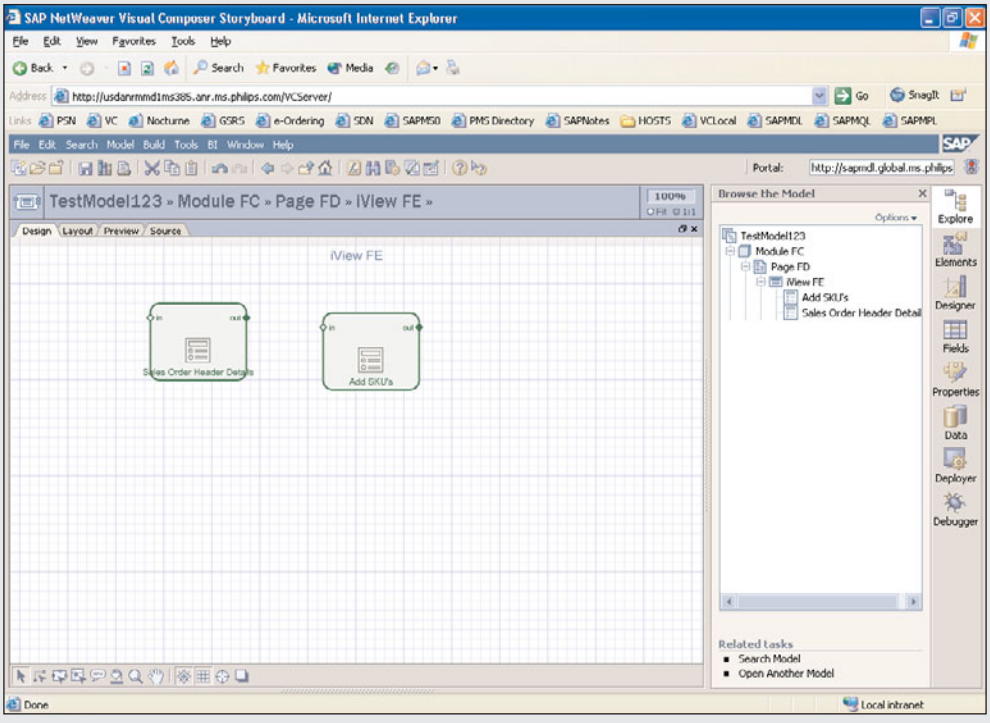


Figure 3 Visual Composer Design workspace

simplification of an SAP transaction is your main reason for customization.

- **Business packages** are composed of prefabricated portal code that you can download from the Portal Content Studio on the SAP Developer Network (SDN). The code is not modifiable, but at a minimum, it provides a good example of how your custom interface can and should look. **Figure 2** shows one of the many business packages available for download from the Portal Content Studio.[1]
- **Visual Composer** is an exciting development from SAP. Visual Composer is a Web-based workbench that lets business developers develop interfaces with no coding for both SAP R/3 function modules (RFC and BAPI) applications and SAP Business Information Warehouse (SAP BW) report applications. **Figure 3** shows the Visual Composer Design workspace. You can drag the design elements from the list on the right and drop them into the workspace to model your custom project. **Figure 4** shows the resulting layout of your project. Each of these workspaces and its tools give you the flexibility and vision to customize your project.

One thing to be aware of with Visual Composer is that the developed interfaces are usually hosted as iViews in an SAP Enterprise Portal (6.0 SP3 and higher). This isn't all that surprising since SAP is using portals as the presentation component for more and more applications (e.g., ESS, MSS). I really like this approach because a programmer can develop an iView in minutes. The main disadvantage to this approach is that the format and type of rendering are currently limited (for example, alerts cannot be constructed), but in the next version of

[1] Go to www.sdn.sap.com/sdn/index.sdn for more business packages.

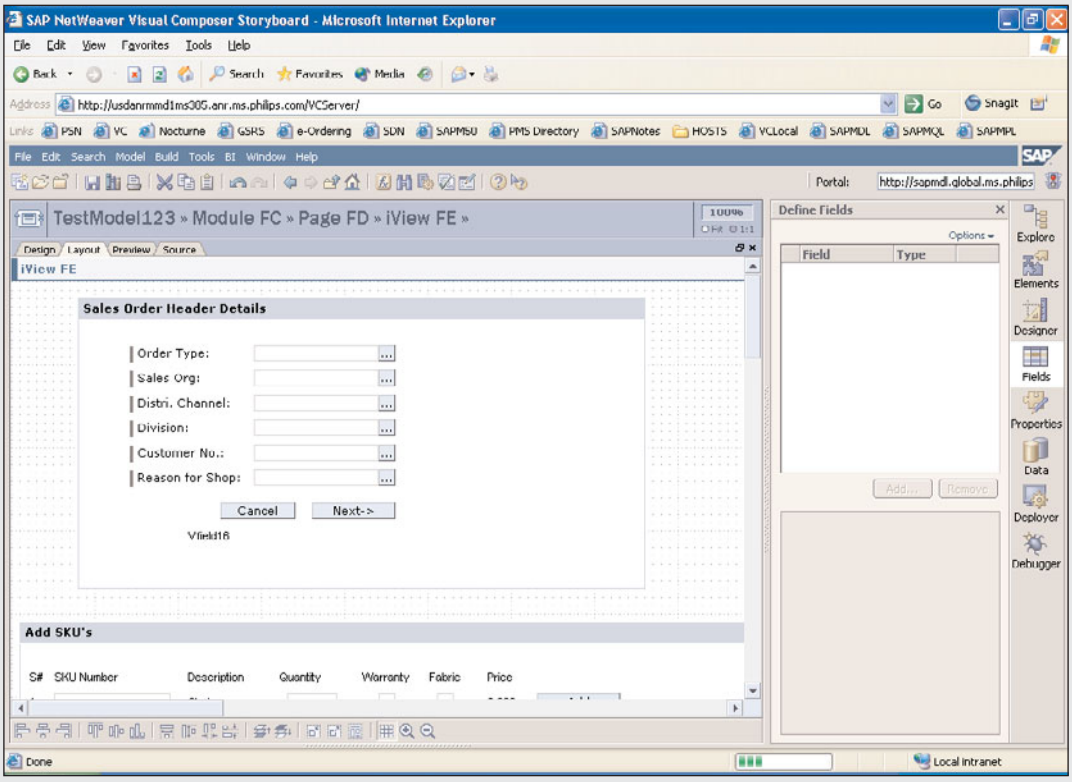


Figure 4 Visual Composer Layout workspace

Option	Advantages	Disadvantages	Used by
Business packages	Generally well designed	Non-modifiable	All users (can be a good starting point)
GuiXT	Allows quick manipulation of SAPGUI and ITS screen fields	Works only with SAPGUI and ITS screens and cannot add new fields	Business application developers and ABAP programmers
Business Server Pages (BSPs)	Uses ABAP	Initial rendering can be slow	ABAP programmers
Visual Composer	No programming required	Limited flexibility	Business application developers
Web Dynpro	Can be used for both SAP and non-SAP applications	Knowledge of Java needed	Java programmers
Portal Development Kit (PDK)	Makes use of portal-specific Knowledge Management (KM) classes	Web Dynpro can do the same for non-KM applications	Java programmers
PDK for .NET	Used for Microsoft applications	Cannot be hosted by SAP Web Application Server (SAP Web AS)	Active Server Page (ASP) programmers
Native development (.NET Connector or Java Connector)	No new tools needed for Java or .NET developers	No shortcut to make use of portal functionality such as branding (style sheets)	Java and .NET developers with no knowledge of SAP Enterprise Portal

Figure 5 A sampling of available tools for custom development

Visual Composer, SAP has promised significant enhancements, such as drop-down lists to simplify data entry and integration with robust Internet applications, such as Macromedia's Flex.

Understand both the advantages and disadvantages of a particular tool set

If you determine that custom coding is your only alternative, you need to understand what tools are available and how they work. **Figure 5** summarizes the advantages and disadvantages of tools with which I am familiar.

Business packages provide a good starting point for custom development. The source code is generally not available with the business package, but the screen layouts provide a good model for the development of custom screens. GuiXT is also a good way to model customized SAPGUI and ITS screens.

For ABAP installations, Business Server Pages (BSPs) are a good way to introduce your programmers to Web development because BSPs contain HTML wrapped around ABAP. This means that ABAP developers can create custom Web interfaces with minimum Web skills (and as a bonus, they'll pick up new skills in the process). **Figure 6** shows how a BSP is rendered in a browser.

You can stick with ABAP for custom development of SAP Web and non-Web screens, but there are cases where that might not be your best option. When considering the construction of Web-based custom interfaces, you must remember that ABAP (even when used in BSPs) is still resident on the back-end SAP system, which leads to a lag in the initial rendering of the Web page. A great alternative for ABAP shops is to develop RFCs and BAPIs and then use Visual Composer to render the interface into an iView for display via SAP Enterprise Portal.

With the introduction of the SAP NetWeaver platform, SAP has made significant tools available to

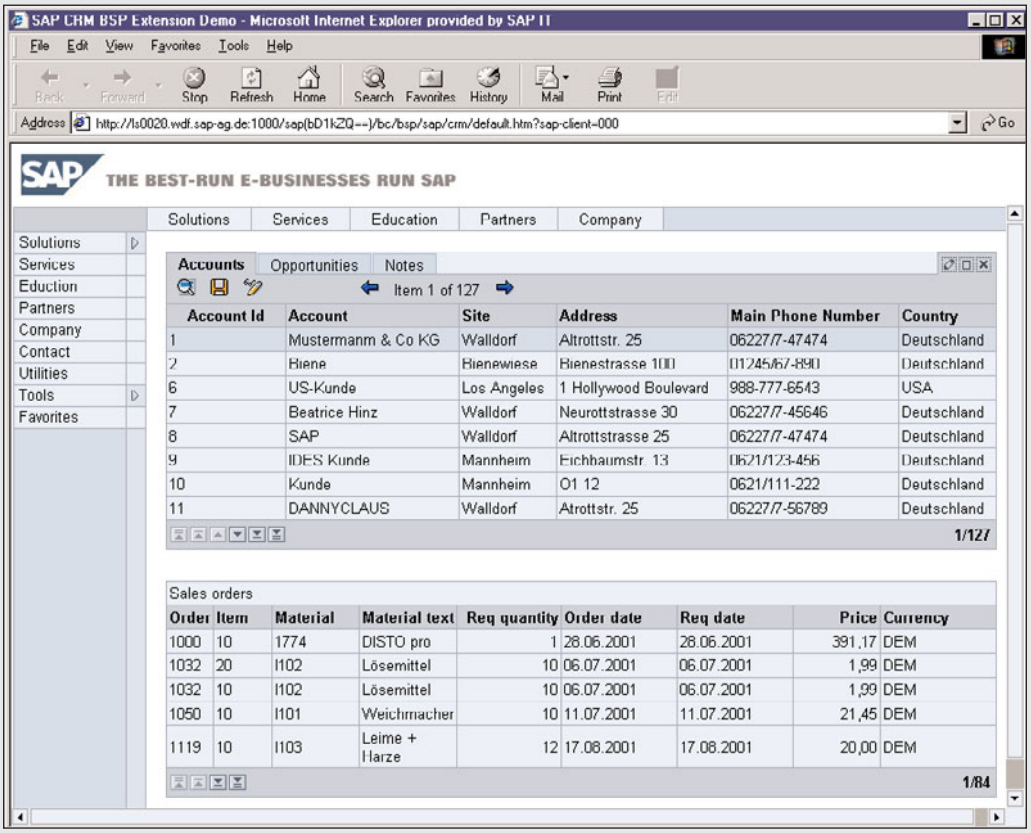


Figure 6 Business Server Page (BSP) iView

developers.[2] For expert Web developers, SAP has provided SAP NetWeaver Developer Studio, which enables the development of applications using Web Dynpro, which is essentially a Web-native version of the well-known Dynpro programming model that uses Java rather than ABAP. SAP NetWeaver Developer Studio is a highly visual tool, but it does require some technical knowledge and a good knowledge of Java.[3]

[2] For more information on custom development on the SAP NetWeaver platform, read the *SAP Professional Journal* articles "Developing Custom Applications for SAP Enterprise Portal — Starting with the 'Right' Options in Light of SAP NetWeaver" (March/April 2005) and "Developing Custom Applications for SAP Enterprise Portal — Advanced Java and .NET Options to Consider in Light of SAP NetWeaver" (May/June 2005).

[3] For a detailed introduction to this tool, see the article "Get Started Developing, Debugging, and Deploying Custom J2EE Applications Quickly and Easily with SAP NetWeaver Developer Studio" (*SAP Professional Journal*, May/June 2004).

For existing Java installations, my preferred option is to use SAP NetWeaver Developer Studio to produce Web Dynpros. Web Dynpros can be written for non-SAP applications, but they really show their power when used to interface to SAP applications. Web Dynpros are true front-end Web applications that are hosted by SAP Web AS 6.40. Because they are hosted applications and not back-end SAP applications, the render time is very good.

SAP has also developed a transport mechanism for Web Dynpro — the Java Development Interface (JDI) — that allows versioning and transport control in line with what currently exists for the SAP back end. (Until the advent of the JDI, versioning and transport control were a major hindrance to Java development.)

You may be surprised that I only briefly mentioned the Portal Development Kit (PDK) in Figure 5. I haven't

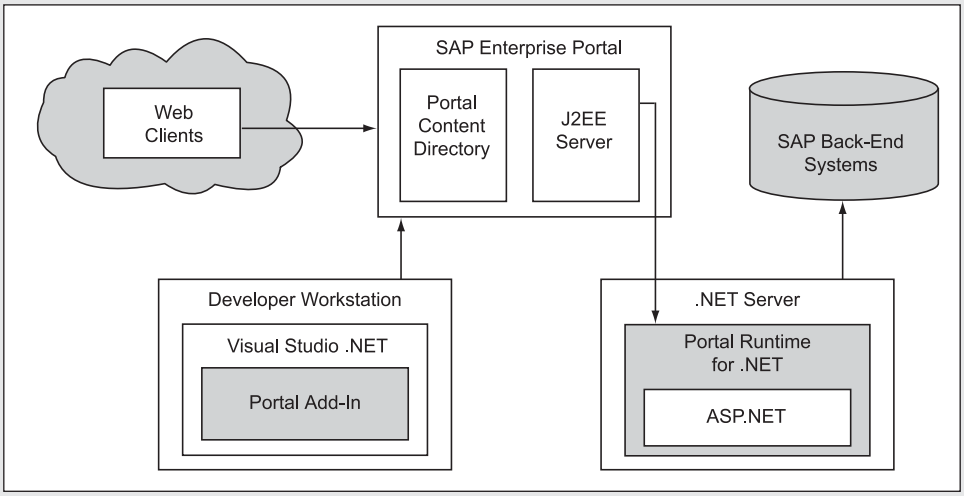


Figure 7 Technical system landscape for PDK for .NET

focused on it because, in my opinion, Web Dynpro can effectively replace all the functionality provided by the PDK for Java iViews. The PDK is a halfway house between pure Java and workbenches such as Eclipse, JBuilder, etc., and it includes Java classes to access SAP native functionality and to construct objects such as scrolling tables.

Do not discard the PDK completely, however, because there is one area in which the PDK is very useful: accessing and developing functionality associated with SAP Knowledge Management (KM), because the PDK has many KM-specific Java classes.

In addition, a PDK designed specifically for the .NET world — PDK for .NET — sits on top of Microsoft Visual Studio (see **Figure 7**). This product is a valuable option for the Microsoft Active Server Page (ASP) developer because there is no Web Dynpro equivalent for developing .NET applications. PDK for .NET provides a library of SAP-specific .NET controls that allow the developed applications to make use of portal services and themes.

Lastly, if you are a Microsoft shop and you do not want to use PDK for .NET, you can leverage your development resources by using one of the SAP connector products, such as .NET Connector or SAP Java Connector (JCo). With this approach you can write applications that interface with your SAP back-end systems. Using these components, you can code the front end either in native Microsoft languages, such as ASP, or in open source, such as Java. The disadvantage of the native approach is that the applications do not naturally make use of the SAP style sheets or functions that allow components such as scrolling tables to be automatically constructed. In addition, they are not SAP NetWeaver-compliant because the developed application needs to be resident on a Microsoft Web platform such as Internet Information Server (IIS), so two discrete environments have to be maintained with all the administrative and cost implications involved.

Match the available customization tools to your in-house skill set

Once you understand the tools you have at your disposal, you must determine which are the most suitable for your organization. Look at your existing in-house skills (to make use of what you have). For most SAP shops, this will be ABAP, but for newer SAP installations, it could be Java or Microsoft.

Choosing the "best" option requires you to carefully consider the following:

- The predominant skills of the team members in your organization
- The platforms you have available (or plan to upgrade to)
- The systems you need to access
- The complexity of your custom application
- The amount of customization required
- How your landscape (and available technology) will evolve in the future

Business packages and tools such as GuiXT provide "light" custom development options; however, for more heavy-duty customization, you have to look at Web Dynpro and BSPs.

In the past, when I've had clients developing all their custom applications in ABAP, I would try to lead them to develop interfaces in BSP. Recently, however, I have been moving away from this path because SAP seems to be on the brink of discontinuing BSP development — for example, as of SAP Enterprise Portal 6.0 SP3, the Theme Editor tool refers to BSP style sheets as "old" instead of as "BSPs," which was the original wording.

For complex one-off interfaces, the use of a third-party Java programmer can be very helpful; however, if a large number of custom interfaces are required, I recommend training in the use of SAP NetWeaver Developer Studio to create Web Dynpros. Either select someone with an ABAP background — because this person will understand both the front-end and back-end development — or use a Java developer who works with the ABAP team to understand the interfaces and functionality of the back-end programs.

Understand the future direction of SAP development

To ensure the longevity of your coding efforts, it's important to consider the direction in which SAP is heading. SAP appears to be heading toward an open architecture — and that means Java — but also toward a world in which the business user can design an application with little coding knowledge.

You probably noticed that I did not select Visual Composer as either my light- or heavy-duty option for custom development. This is because it currently fits in-between both options; however, from the demonstrations that SAP performed at SAPPHIRE 2005, it seems that the next version of Visual Composer will be able to provide much of the functionality that we can currently get only with Web Dynpro. Indeed, SAP seems to be moving to a common platform with both Visual Composer and Web Dynpro, a platform in which code will be interchangeable between the two development tools. This means that a Web Dynpro programmer will be able to enhance the presentation layer produced by Visual Composer within SAP NetWeaver Developer Studio. Conversely, the business developer will be able to alter Web Dynpro presentation using Visual Composer.

Conclusion

Custom code tends to live longer than most people think, and which options are strategic can change over time. If you've got too much custom code, the road to recovery starts by implementing defined initiatives to reduce the number of platforms and technologies you use and to consolidate and harmonize your existing custom code. SAP NetWeaver can help with that.

I see two directions that custom development is going to take:

- Custom applications produced by business users, using a development tool that requires no coding by the developer, such as Visual Composer
- Custom applications produced by an integrated workbench to allow all code to be built and deployed centrally, such as Web Dynpro

This article explored the options that are available to you for your custom development needs and provided a guide for matching those options to both your existing skill sets and your future development strategy so that you can bring both your developers and users into the

SAP NetWeaver world. SAP is trying to prevent the new SAP NetWeaver custom development tools from making existing SAP development skills — in particular, ABAP — obsolete by promising to include ABAP in new versions of Web Dynpro (Web Dynpro for ABAP).

One last point: You do not need all your systems to be on SAP NetWeaver to run SAP NetWeaver interfaces. It is only the interface that needs to be located on an SAP NetWeaver platform — in other words, deployed on SAP Web AS 6.40.

Web Dynpros, for example, can be written to access data from any SAP system, whether the system is SAP NetWeaver or not. This is good news because most organizations will not convert to SAP NetWeaver overnight, but instead gradually update components of their SAP infrastructure to SAP NetWeaver. Many of my clients, for example, moved to SAP BW 3.5 first to "test the waters" and are now developing migration plans for their SAP R/3 systems. The new SAP NetWeaver development tools are strengthening their belief that moving to a homogeneous SAP NetWeaver environment is the correct path to take.

