Scenario 1: Patchwork and niches

Customizing

An organization licenses and implements standard software and customizes that software with the help of the customization tools available for the software, e.g. parameterization or model-based generation. Missing features are added via APIs (application programming interfaces). This means that some parts of the overall solution are developed specifically for the individual organization, either inhouse, by a software firm, or by the vendor of the standard software.

Extending standard software

If an entire problem area relevant to the organization is not covered by the standard software, then a complete information system will be developed either inhouse or by a partner (see scenario 4), or purchased from a different vendor. This new system has to fit the rest of the company's information systems, not only regarding technology but also integration on a logical level. Restrictions and requirements for the new IS are set by the organization's information systems architecture. If the new system does not match those restrictions and requirements, bridges have to be built to make the new and the existing systems compatible ("bridge programming"). Connecting software products from different sources or vendors can be a non-trivial development problem.

New releases of the standard software may create problems for add-on systems so that those systems need to be modified.

Scenario 2: Personal information management

Definitely not all tasks at all individual workplaces in an organization are supported by standard business information systems, yet most people today use a personal computer for their daily work. Almost all white-collar workers do, as do many blue-collar workers as well. Many workplace related tasks, on the personal level, are solved with the help of office programs.

Development by end-users through visual tools and IDEs

While simple problems may be addressed using those tools directly, more complicated tasks require the development of programs (often called "macros") or entire information systems. With end-user oriented tools and languages, workers can develop themselves, or have someone develop for them, quite powerful information systems based on Microsoft Excel and Access, for example. Excel and Access support end-user development through visual tools and the VBA (Visual Basic for Applications) language. As the level of IT education in the business world is increasing, adept professionals may also develop larger solutions for

their individual tasks in a convenient IDE (integrated development environment) such as Visual Studio .NET.

A typical example is data analysis

Typical examples of personal information management are end-user systems for *data analysis*. Using data provided by one of the standard business systems, an information system in Excel based on pivot tables may be employed at the workplace to analyze the data and create nice-looking charts. Many ERP, SCM, and CRM systems today provide download and upload features and interfaces for office programs such as Excel, Access, and Outlook. SAP and Microsoft even developed a common software (called Duet; http://www.duet.com) to enhance integration of SAP application software (such as SAP ERP) with personal information management (based on MS Office Professional).

Software organizations

By *software organizations* we denote professional software firms that live off producing and selling software, as well as IT departments, software development groups and subsidiaries of large organizations whose primary task is to produce and maintain software for their parent organizations. A special type of software organization, with blurry edges, are loose networks of developers that create open-source software.

Scenario 3: Large-scale development

Entire information systems such as ERP, SCM or CRM systems are usually created by organizations that have the financial power to invest large amounts of money in standard software development and receive the returns only after some years.

Development of standard software rarely starts from scratch

Although this type of development is not constrained by an existing inhouse IS landscape like a user organization has, development rarely starts from "nothing is there" either. Unless the problem domain is a completely new one, some older system or at least some modules are likely to exist already. If an ERP vendor, for example, decides to set up a new ERP system, then that company probably has experience in the field from selling such a system. Since not all parts of the old system will be obsolete – a bill-of-materials processor, for example, will always process bills of materials in the same way – some of the old program code is likely to survive into the new system.

Existing systems limit the degree of freedom

Upgrading an existing system is more common than developing an entirely new system from scratch. New versions or releases are produced based on the existing system, adding new features and revising and improving old features. This limits the degree of freedom rather heavily.

The new release has to run in the user organization's social and technological environments, often in identical environments as the old releases.

New systems and releases are often subject to the constraint that interfaces for industry-standard systems in adjacent areas have to be provided. An ERP system, for example, needs interfaces for Siebel CRM, and any non-ERP system will require interfaces for SAP ERP.

Scenario 4: Custom information systems

Although really large information systems are usually standard software (scenario 3), this software may need to be substantially extended for an individual organization. Likewise, entirely new, individual information systems may be needed to fill gaps not covered by the standard software. These types of development are often contracted to professional software organizations. That organization will carry out the development, working together with employees of the user organization at various levels and stages.

Software company and user organization working together

If the user organization has their own IT staff, a typical division of labor is that these people do the requirements engineering and produce a requirements specification document (cf. section 2.2.1). The software vendor will design the system, develop it and deliver it to the customer. There it will be tested and evaluated by business and IT staff.

Scenario 5: Open-source development

Open-source software (OSS) is software that is available as source code to the public free of charge. All types of software exist as open-source: operating systems, database management systems, web servers, office programs, and even business information systems such as ERP and CRM systems.

Sources of OSS systems

The development of OSS comes in many variations. OSS is typically developed around a nucleus – a software system – that was initially created by an organization or individual and then made available to whoever is interested in the code. Many developers around the world revise the code and contribute additional code. Some OS systems started out from hobbyist programming by individuals who wanted to do good to the world (or perhaps bad to capitalist organizations exploiting the world through costly software). Other OSS was initially created by professional organizations and then made available to the rest of the world. The primary reason for doing so is normally not altruism but to earn money from services and software based on the OSS.

Open-source development is incremental and iterative

Open-source development goes on in an incremental and iterative way. Since many people or organizations are involved, revising and contributing code, there are usually rules of how new versions of the software become public – sometimes strict rules, sometimes loose rules, sometimes practically no rules. Nevertheless, anyone interested in OSS may download the code, use it, incorporate it in their own systems, and build new systems based on that code. Legal obligations may have to be fulfilled when revenue is earned from the new systems (see section 4.6).

Outlook

All five scenarios are simplified abstractions of real-world situations, yet they are useful to distinguish different types of information systems development.

Scenario 4 is the one that comes closest to what was underlying classical ISD. Scenario 2 will not be covered in this book. Scenario 1 is the dominating scenario for most user organizations today. Many of the methods and tools discussed in this book apply to large-scale development by software organizations (scenarios 3 and 4) as well as to development around standard software (scenario 1).

While pure OSS projects (developing open-source software within the OSS community) are not the focus of this book, the use of OSS systems or modules within large-scale professional systems (scenario 3) and within custom information systems (scenario 1) is an increasingly important aspect.