**CH2. The Origins of Software**

* 說明outsourcing.
* 描述 six different sources of software.
* IT services firms
* Packaged software producers
* Enterprise-wide solutions
* Cloud computing
* Open source software
* In-house developers
* 討論如何評估 off-the-shelf software.
* Explain reuse and its role in software development.

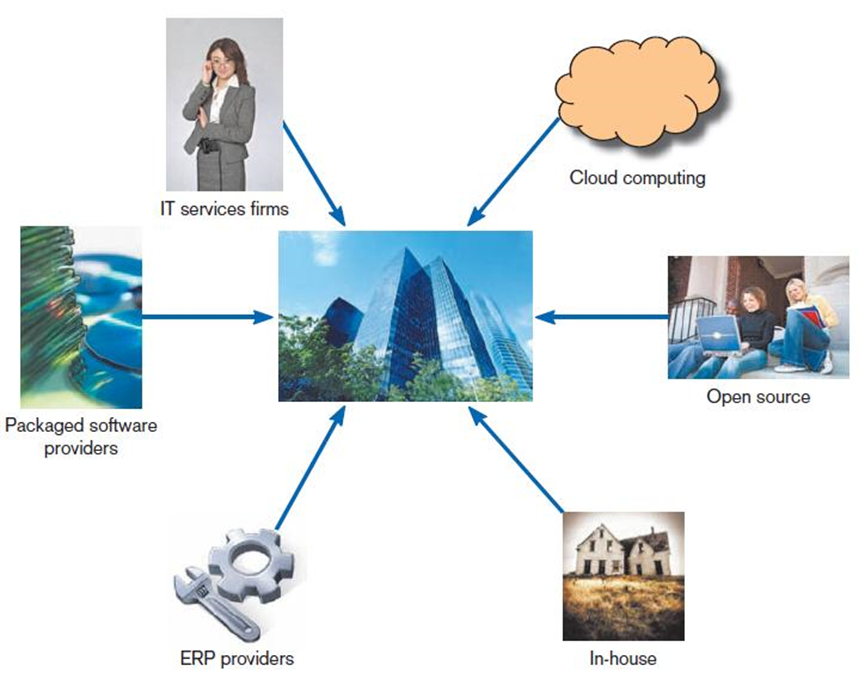
1. **Systems Acquisition: Outsourcing**

Outsourcing: Turning over responsibility of some or all of an organization’s IS applications and operations to an outside firm

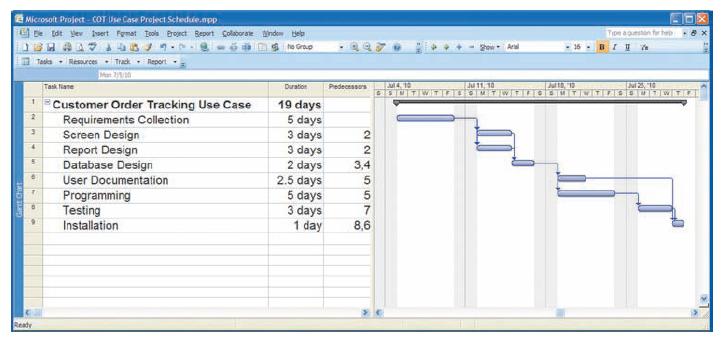
Ex. Shell Oil -> EDS, T-Systems, AT&T

* **Reasons to outsource**
* Cost-effectiveness
* Take advantage of economies of scale
* Make up for lack of in-house knowledge
* Free up internal resources
* Reduce time to market
* Increase process efficiencies
* System development is a non-core activity for the organization

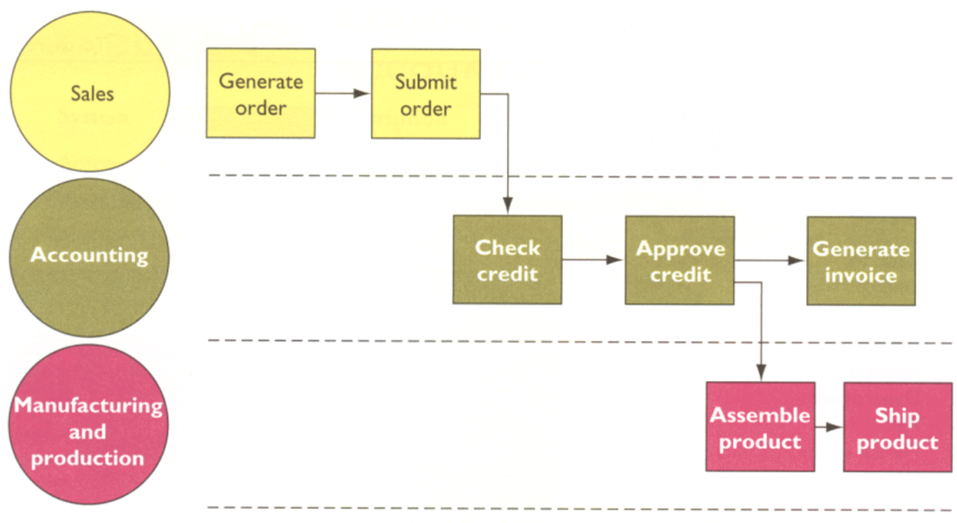
1. **Sources of Software**

****

* **IT services firms**
* Help companies develop custom information systems for internal use
* Develop, host, and run applications for customers
* Provide other services
* **Packaged software producers**
* Serve many market segments
* Provide software ranging from broad-based packages (i.e. general ledger) to niche packages (i.e. day care management), running on different platforms
* Pre-packaged, off-the-shelf software
* Off-the-shelf software, at best, meets 70% of organizations’ needs.



Microsoft Project(上圖)

* **Enterprise-wide solutions**
* Many firms have chosen complete software solutions, called enterprise solutions or Enterprise Resource Planning (ERP) systems
* ERP systems integrate individual traditional business functions into modules enabling a single seamless transaction to cut across functional boundaries.
* A single repository of data for all aspects of a business process and the flexibility of the modules
* Focus on businesses rather than on business functional areas

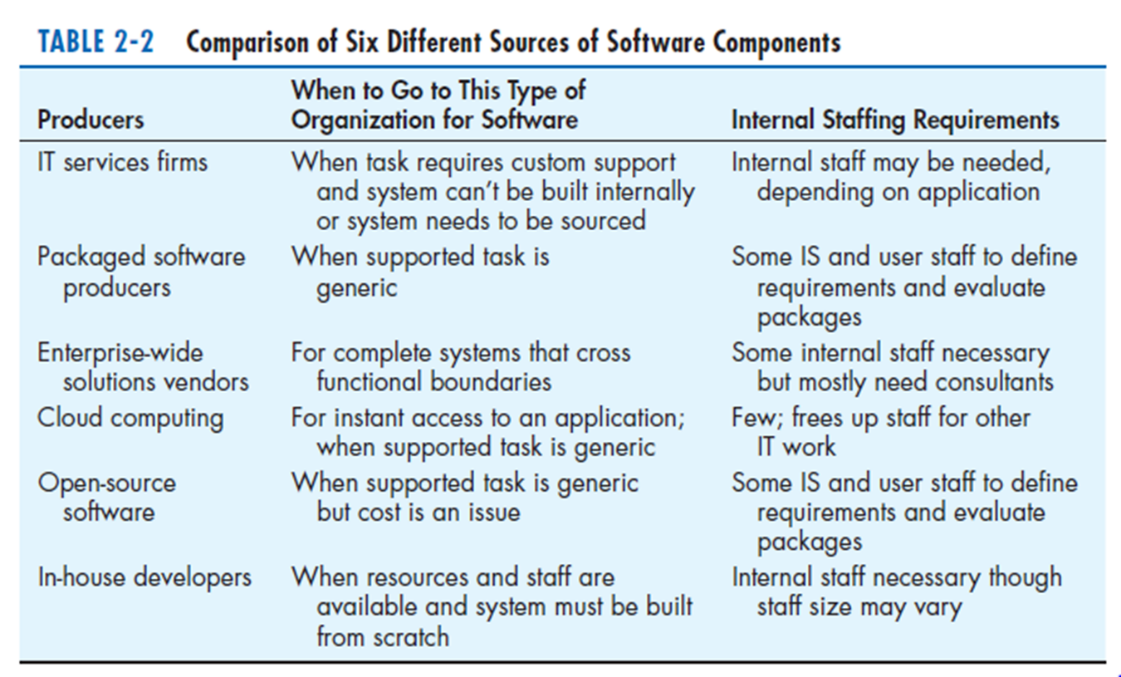
Order Processing下圖

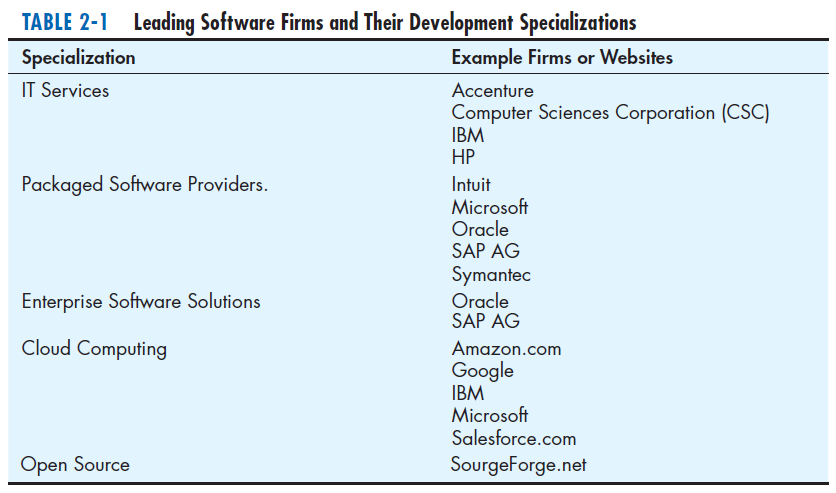
* **Disadvantages**
* Complex: implementation can take a long time to complete
* Expensive: reliance on consultants or employees of the software vendor
* Changes of business processes in some cases



SAP’s Business ByDesign, for medium sized companies.

* **Cloud computing**
* The provision of computing resources, including applications, over the Internet, so customers do not have to invest in the computing infrastructure needed to run and maintain the resources
* Organizations rent apps or license apps from 3rd-party providers who run the apps at remote sites
* Users have access to the apps through the Internet or through VPN
* The app provider buys, installs, maintains, and upgrades the apps
* Users pay on a pay-use basis or they license the software, typically month to month
* Examples:
* Google Drive – users store what they want on servers (hardware as a service, HaaS)
* Salesforce.com – online customer relationship management (CRM) software (software as a service, SaaS)
* Microsoft Azure platform (platform as a service, PaaS; infrastructure as a service, IaaS)
* Amazon.com cloud infrastructure and services
* **Benefits**:
* Free internal IT staff
* Lower cost and faster access to application than via internal development (an expensive and time-consuming process)
* Speed to market and better performance
* **Concerns**
* Security
* Reliability
* Regulation compliance
* 3 steps process for secure migration to cloud
* Have the company’s security experts involved early
* Set realistic security requirements
* Do an honest risk assessment
* **Open source software**
* Freely available including source code
* Developed by a community (e.g., SourceForge.net) of interested people
* Performs the same functions as commercial software
* Examples: Linux, mySQL, Firefox
* How to make money?
* Provide maintenance and other services
* Sell a more featured version of the free software
* **In-house developers**
* A larger maintenance burden than other development methods
* If sufficient system development expertise with the chosen platform exists in-house, then some or all of the system can be developed by the organization’s own staff.
* Hybrid solutions involving some purchased and some in-house components are common.

****

****

1. **Selecting Off-the-Shelf Software**

* **Cost**:

comparing the cost of developing the same system in-house with the cost of purchasing, licensing the software package, upgrades or annual license fees, maintenance

* **Functionality**:

the tasks that the software can perform and the mandatory, essential, and desired system features

* **Vendor support:**

whether and how much support the vendor can provide and at what cost

* Installation, training, and provisions of help as problems arise
* **Viability of vendor**:

can vendor continue to adapt/update software to changes in systems software and hardware

* **Flexibility**: the ease with which software is customized
* **Documentation**: understandable and up-to-date user’s manual and technical documentation
* **Response time**: how long it takes the software package to respond to the user’s requests in an interactive session
* How long it takes the software package to complete running a job
* **Ease of installation**: a measure of the difficulty of loading the software and making it operational

1. **Validating Purchased Software Information:**

* Collect information from vendor
* Software documentation
* Technical marketing literature
* Information provided upon request
* **request for proposal (RFP)(徵求建議書)**

a document provided to vendors to ask them to propose hardware and system software that will meet the requirements of a new system.

* **Request For Quote** (**RFQ)(詢價)**

vendor bids, a variety of information sources

1. **Information Sources**

* Vendor’s proposal
* Running software through a series of tests
* Feedback from other users of the vendor’s product
* Independent software testing services (e.g., Auerbach Publishers and DataPro)
* Articles in trade publications

1. **Reuse**

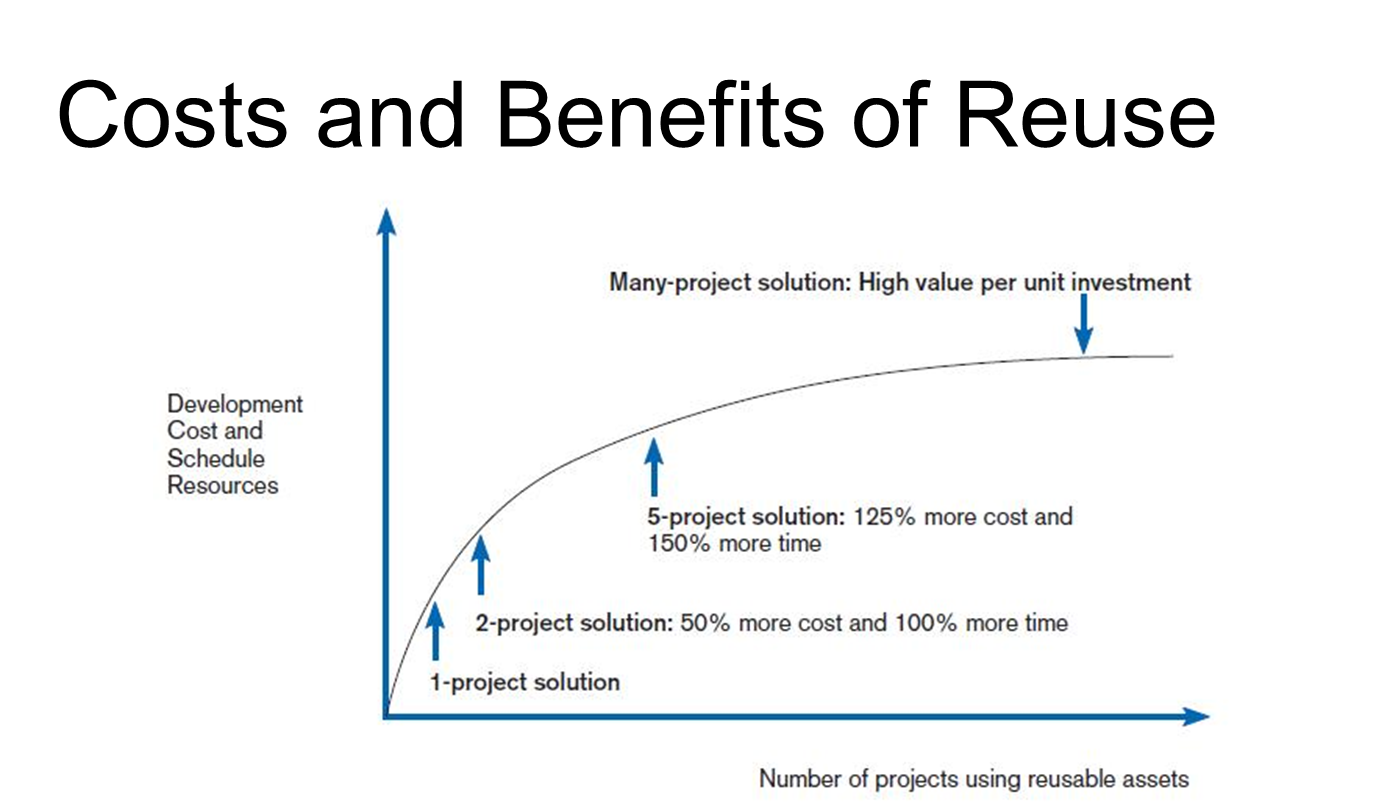
* The use of previously written software resources, especially objects and components, in new applications
* Increase productivity, decrease development time (minimize schedule overrun), result in higher-quality software with lower defect rates (decrease maintenance cost)
* Commonly applied to two different development technologies:
* **Object-oriented development**

Object class encapsulates data and behavior of common organizational entities (e.g. employees)

* The use of object classes in more than one application
* **Component-based development**

Components can be as small as objects or as large as pieces of software that handle single business functions

* the assembly of an application from many different components at many different levels of complexity and size (e.g. Currency conversion).
* **Costs and Benefits of Reuse**



* **Approaches to Reuse**
* **Ad-hoc(特定)**: individuals are free to find or develop reusable assets on their own
* Not really an approach at all
* Few organizational rewards
* Storage is not an issue
* **Facilitated**: developers are encouraged, not required, to practice reuse
* Some tools and techniques are available for developing and sharing reusable assets
* Some may be assigned the role of evangelist(傳教士)
* Little is done to track the quality and use of reusable assets
* Overall corporate investment is small
* **Managed**: the development, sharing, and adoption of reusable assets is mandated
* Processes and policies are established to ensure that reuse is practiced, that the results are measured, that the quality of reusable assets are good
* The focus is on identifying existing assets from various sources
* **Designed**: assets mandated for reuse as they are being designed for specific applications
* The focus is more on developing reusable assets than on finding existing assets
* A corporate reuse office may be established to monitor and manage the overall methodology

