



INFORMATION TECHNOLOGY MARKETING

BY: Meisam Hejazinia

OUTLINE-1

- Terminology of Software Business
- Software Business Ecosystem
- Software Business Functions
- Software Industry Analysis
- Software Business Strategy

OUTLINE-2

- Software Business Operations
- Software Business Entrepreneurship
- Open Source Software Initiation
- Open Source Business Models and Revenue
- Open Source Software Operations
- Commercial Open Source

OUTLINE-1

- Terminology of Software Business
- Software Business Ecosystem
- Software Business Functions
- Software Industry Analysis
- Software Business Strategy

SOFTWARE PRODUCT: TERMS AND CHARACTERISTICS

- An intangible economics good
- Software belongs not to the three classics economic factors of capital, land, labor but in the new fourth category of “knowledge”.
- It could have great impact on different aspect of organization effectiveness and strategy.



SOFTWARE PRODUCT: TERMS AND CHARACTERISTICS

- product : combination of goods and services, which one party(called vendor) combines in support of their commercial interests, to transfer defined rights to a secondary party(called customer).
- Software product: product whose primary component is software
- Is mobile a software product?
 - Answer: No.

SOFTWARE PRODUCT: TERMS AND CHARACTERISTICS

- Embedded software: a piece of software that is not sold as stand-alone software product , but is integrated in a non-software product.
- OEM software product: software product of company A that is used by company B as a component under the covers of one of B”s product.
- Is search by google a product?

PRODUCT PLATFORM, FAMILY, AND LINE

- **Product platform:** the technical foundation on which several software products are based.
 - A definition for planning, decision making, and strategic thinking.
 - Ex: SAP core system that serves the basis for all SAP component.
- **Product family:** A group of software product which for marketing reasons are marketed as belonging together under a common family name.
 - Marketing products under “family name” is more efficient.
 - Ex: SAP products, IBM’s DB2 family, Microsoft office.

PRODUCT PLATFORM, FAMILY, AND LINE

- **Product line:** A group of software products which are variants of a base product governed by a common software architecture.



PRODUCT NAME, VERSION NUMBERS AND COMPATIBILITY

- Marketing aspects frequently play a more important role than technical aspects in selecting names.
- For example, IBM uses a three-level of software levels:
 - Version
 - Release
 - Modification level
 - An example is IBM z/OS Version 1 Release 9 Modification level 5 (in short z/OS 1.9.5).

PRODUCT NAME, VERSION NUMBERS AND COMPATIBILITY

- In changing from software version n of a product to the next version n+1, existing function of version n continue to be supported.
- Data from version n can be transferred to and used with version n+1 without changes.
- Interfaces of version n(APIs, interfaces for other system/products) remain unchanged.



ATTRIBUTE OF SOFTWARE PRODUCT

- Market:

- Consumer(B2C), e.g. games, software
- Business(B2B)
 - Horizontal (i.e. across many industries): e.g. systems software, middleware
 - Vertical (i.e. industry specific): e.g. securities application for brokerage industry

- Functional areas, e.g. systems software, middleware, application



ATTRIBUTE OF SOFTWARE PRODUCT

- Development focus, i.e. standard software vs. service vs. individual development
- Conditions:
 - Terms of contract, e.g. open source, freeware, shareware, priced licensing, SaaS
 - Development at a fixed price or a price according to effort
- The term “services” has emerged as a new topic as a new topic as the base of serviced-oriented software architecture, often in the form of web services.

ATTRIBUTE OF SOFTWARE PRODUCT

- in every software installation, There are three basic types of software products:
 1. the operating system
 2. middleware
 3. Application
- An important differentiator of products is triggered by market criteria:
 - Business-to-business or B2B
 - Business-to-customer or B2C

SOFTWARE AS BUSINESS

- The term “software”, first coined in a 1958 article by John W. Turkey.
- At NATO conference in Germany in 1968, the term “software engineering” was coined.
- From the late 50s on, the first higher-level programming language like FORTRAN and ALGOL were created.
- IBM announced on June 23, 1969, that it would unbundle hardware and software in the future(this can be seen as the birth date of the software industry).

OUTLINE-1

- Terminology of Software Business
- Software Business Ecosystem
- Software Business Functions
- Software Industry Analysis
- Software Business Strategy

SOFTWARE AS BUSINESS

- Software market:
 - 243 billion \$ in 2007
 - 327 billion \$ in 2010
 - Compound annual growth rate(CAGR) of 7.7%
- IT market:
 - 486 billion \$ in 2007
 - 587 billion \$ in 2010
 - Compound annual growth rate(CAGR) of 5.8%



IT DOESN'T MATTER?

- Industries that are more fundamentally influenced and changed due to IT:
 - Banking and financial services
 - Music
- IT is embedded in a cluster of related innovation:
 - Automation of numerous routine tasks.
 - Highly skilled labor.
 - More decentralized decision making.
 - Improved information flow vertically and laterally.
 - Strong performance-based incentives.
 - Increased emphasis on training and recruitment.

SPECIFIC BUSINESS ASPECT OF SOFTWARE

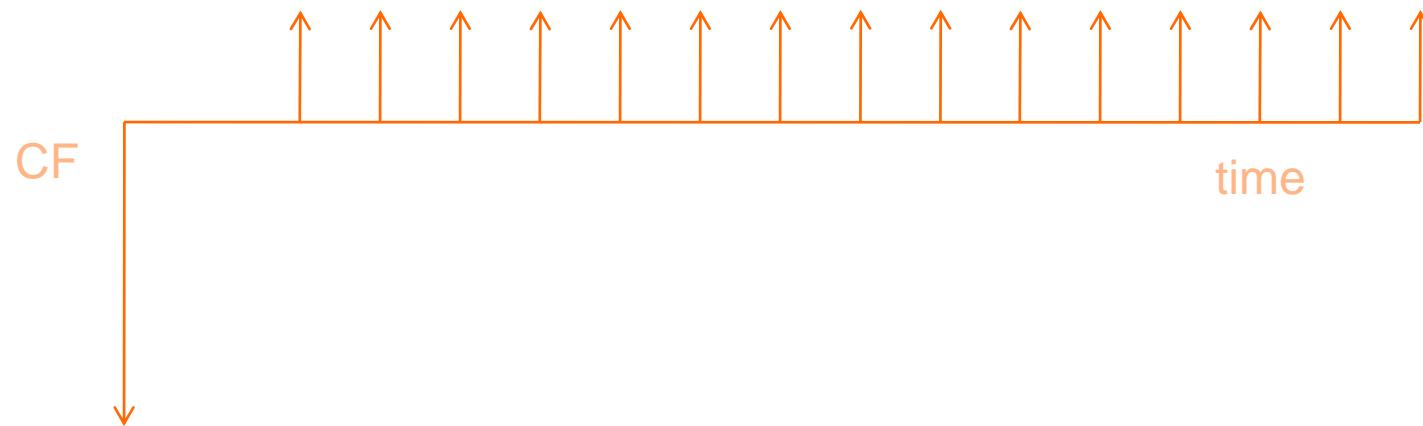
- The fixed cost of development is almost the entire cost of production.
- The variable cost for the single product instance is negligible.
 - Only for low-price consumer software product like PC games variable contribute a higher share of the total cost.
- Ongoing cost of maintenance often charged separately.
- Reaching a certain minimum revenue is mandatory in order to reach the break-even point.
 - High incentive for software companies to offer their products internationally

SPECIFIC BUSINESS ASPECT OF SOFTWARE

- A software company needs little capital and investment initially and overtime.
 - Famous garage start-ups.
 - Low capital investment means low market entry barrier.
 - Software companies are founded daily-and die daily.
 - The significant aspect of software companies is human know-how.
 - This low entry barrier causes extremely high competition and fluctuation.

SPECIFIC BUSINESS ASPECT OF SOFTWARE

- today, software companies prefer services income(steady cash flow not random cash flow):



THE SOFTWARE ECOSYSTEM

- Software Ecosystem: informal network of(legally independent) units that have positive influence on the economic success of a software product and benefit from it.
- Three principle roles participants in a business ecosystem can play:
 1. Keystone:
 - A benevolent hub in the network that provides benefits to the ecosystem and its members.
 - Examples are Sun, IBM, or eBay.
 2. Dominator:
 - A hub that aims at controlling as much space in the network as possible.
 - An example is Apple.
 3. Niche player:
 - Niche players contribute to the ecosystem by covering a specialized area within it in a competent and reliable way.

THE SOFTWARE ECOSYSTEM

- It is usually not part of the responsibilities of a software product manager to decide on the role his company wants to play in an ecosystem. This belongs into the realm of corporate strategy that is owned by executive management.
- the software ecosystem:
 - VADs (Value added distributors):
 - for outsourcing of production and distribution activities, often for enrichment of products with solution components, for management of smaller partners and a better market penetration.
 - VARs (Value added resellers):
 - as extended sales channel, for better market penetration, and sometimes for enrichment of products with solution components.

THE SOFTWARE ECOSYSTEM

- ISVs (Independent software vendors):
 - vendors of application software whose solutions are based on or favor a vendor's own products.
- OEMs (original equipment manufacturers):
 - manufacturers of branded products of their own who imbed vendor code into their product without vendor branding.
- SIs (System integrators):
 - service companies that add solution components and take over the customer-specific installation and customizing up to the overall project responsibility.
- Technological alliances:
 - as sales cooperation, for preinstallation, for completeness of solution offers and synergy in marketing.
- An individual partner can often exist in more than one of these categories.

THE SOFTWARE ECOSYSTEM

- The software vendor gives some part of the revenue and/or profit to its partners in order to grow faster and reach a leading position in the market.



LAW OF INCREASING RETURNS

- A software product with a high market share will experience a further improvement of its market position just because of this high market share and vice versa.
- three main reasons for this law:
 - The network effect:
 - The utility that a given user derives from the good depends on the number of other user who are in the same network as he or she.
 - Increasing cost of switching:
 - Switching from one software to another software is costly because the data created with one program cannot necessarily be digested by the other program

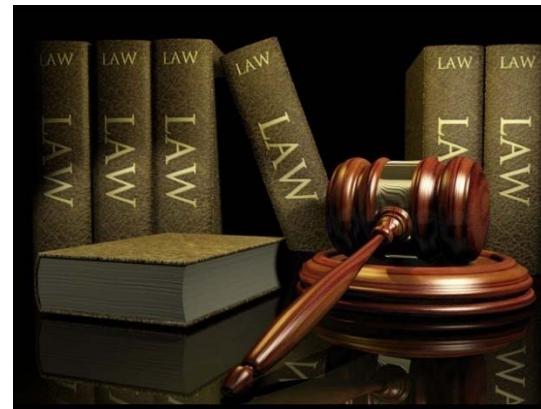
LAW OF INCREASING RETURNS

- Trust in the market leader:
 - Consumers as well as big companies tend to rely on established brands and leading products.
- Because of this law, market leadership is more important for software products than for most other products



LAW OF INCREASING RETURNS

- In an early technology phase there is no market leader.
- During the growth phase of a product market, two or three products and vendors try to become dominant.
- In the maturity smaller vendors disappear and often there is one or two dominant product.



BUSINESS MODELS FOR SOFTWARE VENDORS

- Start-up companies could be financed by venture capital or bank credits or:
 - an alternative is driving development through customer-specific projects that the customer pays for.
- Experience suggests the following rule of thumb:
 - the relation of the effort for a prototype versus a commercially usable piece of software is 1:3, of commercially usable software versus a software product that can be used by a higher number of customers again 1:3.
- Generally we could consider the software vendor model and software service model.

BUSINESS MODELS FOR SOFTWARE VENDORS

- Michael Cusumano:
 - “Regardless of the balance of products and services they choose, managers of software companies must understand what their primary business is, and recognize how the two differ—for selling products requires very different organizational capabilities than selling (professional) services.”
- there are a lot of commercial products that do not cost anything or very little that can differ:
 - Freeware = product available at no charge.
 - Examples are viewer or player products like Adobe’s Acrobat Reader and Flash Player or Real Audio’s player.
 - In these cases, the software vendor intends to make the usage and exchange of files in “his” format easy and at the same time to create demand for his costly full-function-product that is needed to create the files.

BUSINESS MODELS FOR SOFTWARE VENDORS

- Shareware = product available at a price, often downloadable from multiple sites, requires payment of a fee for continued use.
- Trialware = product available either from web download or sometimes media (CD or DVD) which offers a trial period after which it requires payment of a fee for continued use.
- Upgradeware = product where a base version is made available at no charge but where the user is encouraged to upgrade to obtain additional function.
- SaaS offerings financed through advertising.
 - Examples are search engines like Google or Yahoo and Google Apps.

OUTLINE-1

- Terminology of Software Business
- Software Business Ecosystem
- Software Business Functions
- Software Industry Analysis
- Software Business Strategy

BUSINESS MODELS FOR SOFTWARE VENDORS

- Both open source and SaaS offer providing revenue through the support of paid advertisements.
- As an alternative to the model of a software vendor, a company can follow the business model of a professional service company.



OBJECTIVES

- Sustainability is a significant part of any firm's objectives.
- Sustainability is found in a company's assets.



- The precise purpose of product management is to manage systematically such product-related assets on a long-term basis, regardless of whether this involves single products, product families or product platforms.

OBJECTIVES

- Software product management refers to the management of a software product (or product family or platform) over its entire lifecycle in accordance with corporate level objectives.
- Customer satisfaction is often considered a significant measure of software product management success.



THE ROLE OF THE SOFTWARE PRODUCT MANAGER

Day-to day issues

Sustainability related issues

Caretaker

**Software Product
Manager**

Exigencies

Essentials

- The emphasis on sustainability does not imply a company need not quickly and actively take care of urgent customer problems
- The software product manager has a cross-organizational role, requiring a high degree of communication and coordination among all units.
- The scope of responsibility is so extensive that it is hardly possible to find a candidate who can meet all the qualifications

THE ROLE OF THE SOFTWARE PRODUCT MANAGER

- Viewed collectively, software product management can be conceived as neither a project nor a process.



MARKET ANALYSIS

- A software product manager continuously assesses the market based on the inputs received from:
 - Marketing, Sales, Support, Service, and Development
 - Direct contact with members of the ecosystem and other market participants
 - Internal Market Research or Competitive Analysis unit
 - External Market researchers

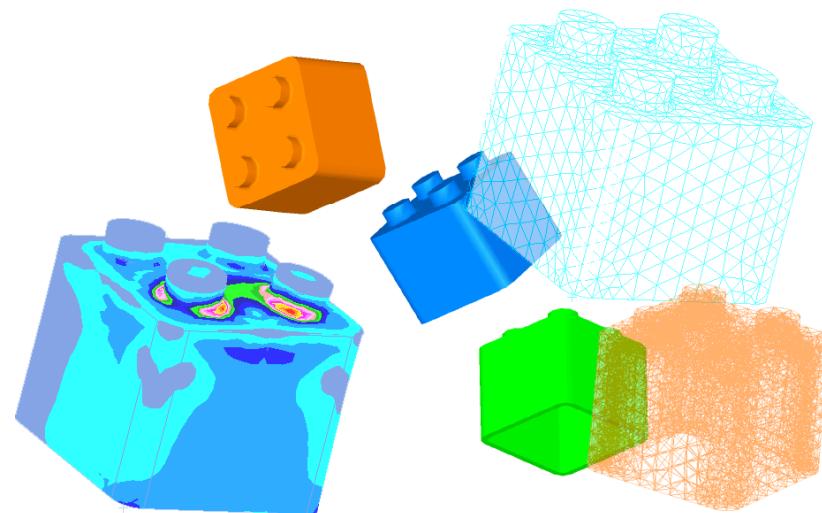


Based on Gartner's Magic Quadrant company's are evaluated and positioned as follows:



PRODUCT ANALYSIS

- The software product manager needs to know at any time how the launched product performs from a business perspective.
- Some important measures (per version, time period, etc.) are:
 - Revenue
 - Cost
 - Profit
 - Number of licenses
 - Defect rate

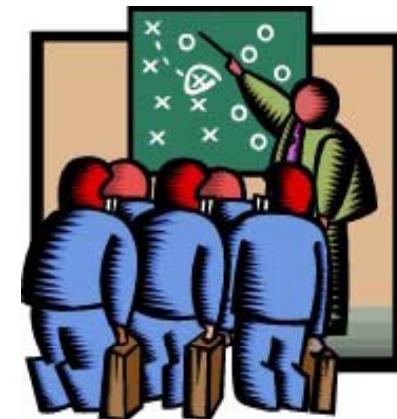


PRODUCT ANALYSIS

- The sources of information for software product manager analysis:
- Customer satisfaction
 - Hard to quantify
 - A multi-variable method
 - Had better be done over an extended period of time
- Feedback from market analyst
- Trade press articles
- Individual customer feedback
- Information from the sales channels
- Information from support and service function

PRODUCT STRATEGY

- Companies with a clear strategy view are the ones that prove to be successful in the long run.
- The software product manager is responsible for defining strategy for his product(s) which includes:
 - Product scope: the approximate functional scope of the product
 - Target market/segments
 - Product delivery model
 - Product positioning
 - Budget and resource planning
 - Roadmap planning
 - Business expectations: development of market share, sales, etc.



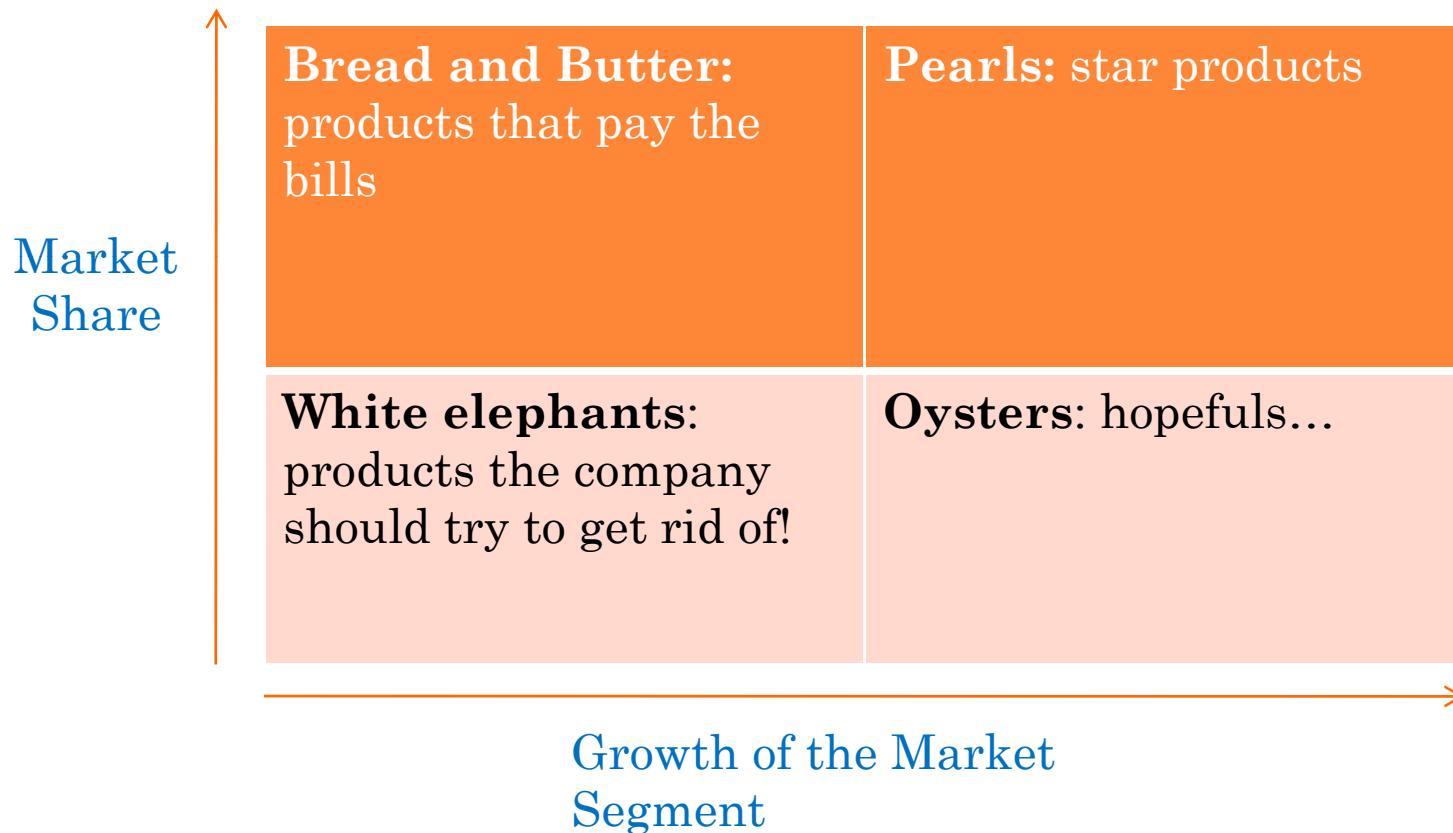
PRODUCT STRATEGY: CORPORATE VISION, STRATEGY, AND PORTFOLIO MANAGEMENT

- One primary task of executive management:
Definition, communication, and implementation of the corporate vision and strategy
 - Requires both stability and flexibility
- The corporate strategy describes how a vision can be implemented
- Portfolio management is the management of the investments over time following profit and tax criteria
 - In portfolio management, not all projects should be put in one single portfolio



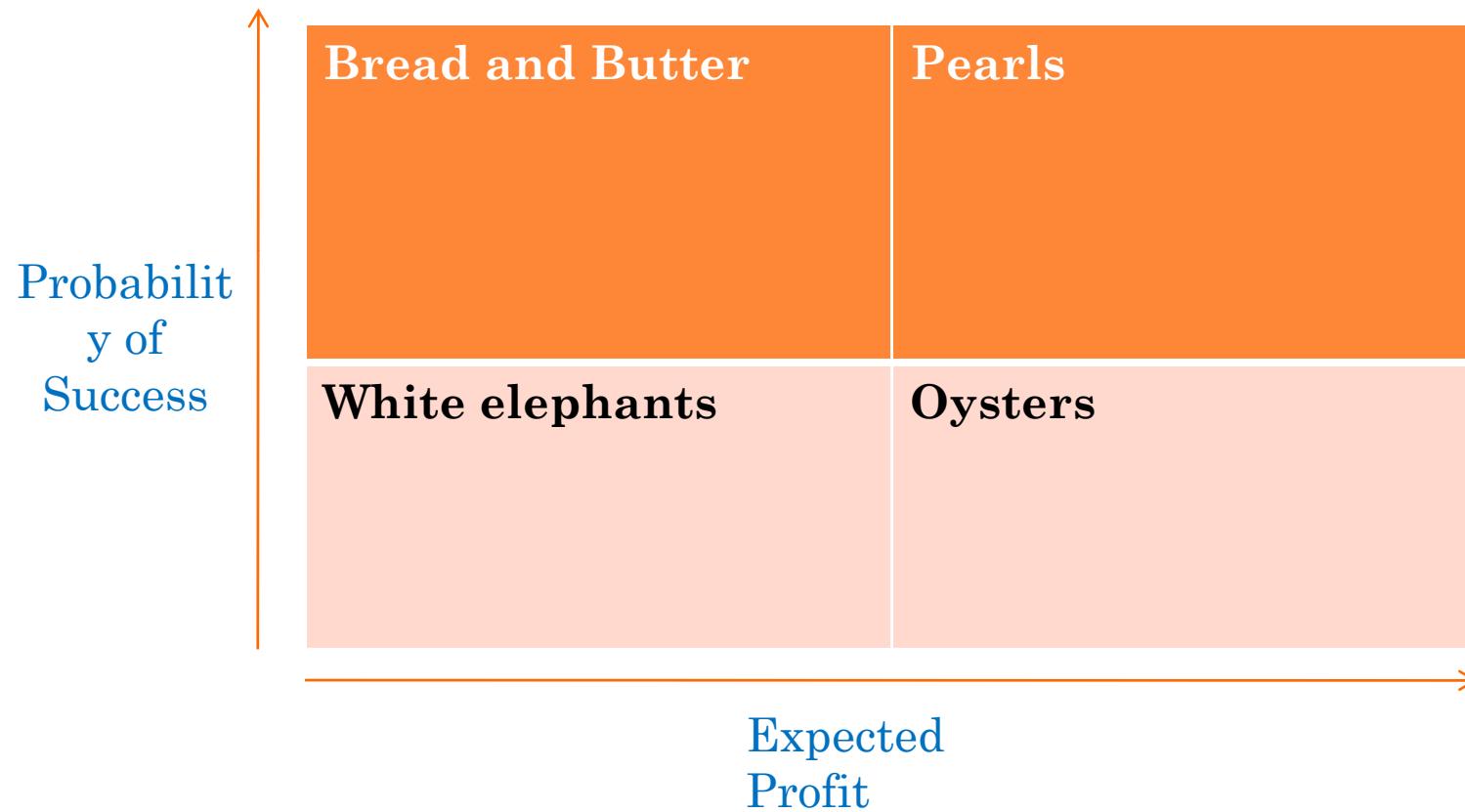
PRODUCT STRATEGY: CORPORATE VISION, STRATEGY, AND PORTFOLIO MANAGEMENT

○ Product Portfolio



PRODUCT STRATEGY: CORPORATE VISION, STRATEGY, AND PORTFOLIO MANAGEMENT

- New Product Development



PRODUCT STRATEGY: CORPORATE VISION, STRATEGY, AND PORTFOLIO MANAGEMENT

- Two important Decisions concerning the new software product offer:
 - The delivery model
 - Make or buy



PRODUCT STRATEGY: CORPORATE VISION, STRATEGY, AND PORTFOLIO MANAGEMENT

- Positioning of the product in the market concerns two important questions:
 - Which market is relevant & how is it going to develop?
 - Where does/should the product play a role?
 - How is the product going to differentiate itself in the market?



PRODUCT STRATEGY: CORPORATE VISION, STRATEGY, AND PORTFOLIO MANAGEMENT

- The Business View:
 - Business case: cost vs. benefit
 - Pricing
 - Product bundle



PRODUCT PLANNING

- Product planning belongs to the core of activities of a software product manager.
- The software product manager is to represent his product(s) in the internal planning process of the company
- Corporate planning is typically a mix of bottom-up and top-down planning



PRODUCT PLANNING

- **Roadmap**

- Gives an overview of how a product going to develop over the strategic timeframe (often up to 5 years)
- Gives internal and external direction

- **Release Planning**

- is concerned with two important questions: When and What
- Small release are faster and more flexible but relatively more costly
- Bigger releases are more economical and often unavoidable

PRODUCT PLANNING

- Requirements Managements and Specification
- Mandatory/Optional Requirements
 - Three dimensions:
 - Customer
 - Product
 - Project requirements management
- The supplementary perspectives on requirements managements:
 - Customer orientation
 - Product orientation
 - Project orientation



DEVELOPMENT

- From a development perspective:
 - All requirements should be defined at the beginning of a project without any subsequent changes.
- From a business perspective:
 - Changes in the market, customer situation, etc. can change the project requirements at any time.
- Development's project requirement management should ensure timely decisions



52

MARKETING

- Marketing is a top priority in the software industry.
- The software product manager influences sales and marketing activities.
- When there is no separate marketing unit within the company, the software product manager has to take care of some or all of the marketing responsibilities.
- Determining the target market is a fundamental prerequisite for defining sales and marketing strategy.



SALES AND DISTRIBUTION

- Just like marketing strategy, the sales and distribution strategy must be consistent with the corporate strategy.
- Appropriate marketing creates the necessary ***pull*** while sales activities provide the supplemental ***push***.
- There are different approaches to sales and distribution:
 - Direct sales
 - Telesales
 - Internet sales
 - Sales and distribution by resellers
 - Channel mix



SUPPORT AND SERVICES

- A key factor in customer satisfaction
- Maintenance services most often comprise 3 levels:
 - 1. Call Center: basic knowledge, differentiate between product failure and user error
 - 2. Dedicated Product Specialist: failure analysis and debugging
 - 3. Developer: particularly difficult problems



OUTLINE-1

- Terminology of Software Business
- Software Business Ecosystem
- Software Business Functions
- Software Industry Analysis
- Software Business Strategy

SOFTWARE BUSINESS

- Not like other business
- Needs to be unique in how they deal with:
 - Business models
 - Product strategy
 - People (software engineers)
 - Management of core activity: software development

ATTRIBUTE OF SOFTWARE BUSINESS

- Same cost of one copy or one million copy
- 99% gross profit margin for product sales
- Product companies eventually become service or hybrid
 - Some customization of product features and technical services
 - such as system integration and maintenance

BUSINESS MODELS

- All or most revenues from new product sales (software license fees)
- Hybrid solutions:
 - 80% of revenues from services and “maintenance”
 - incremental product updates, special enhancement, which have long term contract
- Majority of revenue from
 - IT consulting,
 - custom software development,
 - integration work,
 - technical support,
 - system maintenance,
 - and related activities

EVENTS OCCURRED

- Unwittingly fall into the service or hybrid business models
 - but not prepared for change,
 - for bad economic times,
 - customer postponement in purchase of new products
- Revenues increasingly consist of sales of service
 - and maintenance upgrades to existing customers,
 - rather than new product sales to new customers,
 - since new-software-product sales collapse.

EVENTS OCCURRED

- Potential for rapid change in the market place, software companies must:
 - Combine extraordinary levels of structure with flexibility.
 - Pay constant attention to strategies and business models,
 - as well as continuously evolving their technical skills and core technologies.
- Central to long term success:
 - how software firm manages the technology to create and deliver product and services.

TECHNOLOGY

- Code
- Programming instructions or algorithms
- System architectures
- Program designs
- Data or digital content
- User interface
- Application programming interface
- Programming support and testing tools
- Test cases, documentation
- Other physical and non-physical artifacts
- Effect of what programs do
- How one module of code interacts with other programs
- Modules and computers and what the whole system enables users to do



SUPPLY CHAIN

- Managing the technology in software business (supply chain):
 - overseeing the process of designing
 - a software product or information system
 - for a particular customer need and
 - then building, testing, delivering, supporting, and enhancing
 - that software product or system over its lifetime of use.
- Managing technology should include facilitating communication between
 - scientists,
 - pushing the state of the art and
 - engineers building real-life applications

SUPPLY CHAIN

- Business point of view managing software technology:
 - should imply knowing how to go through the phases
 - from design to delivery at a cost that is
 - less than revenue generated from selling the resulting product or service.
- Marketing and sales and financial discipline become as important as anything else in software business



CONSIDERATIONS

- Need to understand how to build or package technology and price it:
 - Makes people want to buy it
 - e.g. millions of dollar on software development, have little or nothing to show for it
 - Earns a profit
 - e.g. millions on campaign but no gain
- Should concern about flexibility or innovation (e.g. IBM OS/360)
- Chris peters of Microsoft:
 - it is just as important in software development to decide what you are going to do as it is to decide what you are not going to do.

CHALLENGES

- Software can perform an almost infinite number of functions;
 - the challenges of managing the technology well continue to be enormous.
- Demands very considerable depending on:
 - how customers will deploy the software technology in the particular applications or markets.
- “Zero defect” and process excellence:
 - did not necessarily make a company successful in software as a business.
- Japan:
 - more reliable (few bugs or defects),
 - large amount of reused and
 - semi automatically generated code.

JAPAN



- “factory-like” approach,
- incremental innovation in feature design,
- standardized development techniques,
- common training, programs,
- reusable component libraries,
- computer aided support tools,
- rigorous quality assurance techniques,
- statistical data to manage projects.

JAPAN

- Staff projects with a combination of:
 - experienced people and
 - relatively low skilled programmers who had merely a few month of training.
- They don't' know how to make money from software
 - or compete outside Japan.
 - Except software embedded in other products.
- Focused:
 - labor intensive custom or
 - semi-custom information systems build
 - to sell mainframe computers and targeted Japanese customers.



JAPAN

- Had limited:
 - innovation capabilities
 - ambitious.
- Quality, service, reliability: Hitachi
- Dominant in:
 - niches like game industry
 - cause: fascination with comic books from childhood through adulthood



MICROSOFT

- Microsoft managers and programmers did care a lot about:
 - Process
 - Strategy
 - Money
- Software has the power to change the world, especially
 - when treated as a business by managers, programmers, and entrepreneurs
 - who also want to change the world.



Microsoft®

EUROPE

- tend to treat software as a science, while European universities:
 - Formal methods
 - Object oriented
 - Rich but complex applications remarkable in their detail and structure; SAP
 - Simple database query applications and generate report for non-programmers
 - World wide web

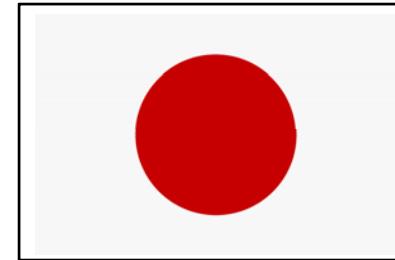


EUROPE

- Many European software producers place more attention on:
 - Achieving elegance in software design than on shipping products for mass markets and
 - Making the most money they can from their excellent technical skills.



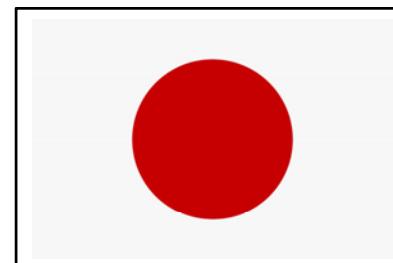
JAPAN



- Significant skill in:
 - writing programs for all sort of application ranging from
 - real-time banking systems to fault free buffet train control and
 - reservation systems, to video game software.
- Most custom-built for specific customers in Japan
- Contain relatively few innovative features, by design
- Like IBM: write code to sell hardware and services
- Country under invested in basic research and higher education
 - weak universities in computer

JAPAN

- Do training software development in production
- Tackled large-scale systems development with
 - heuristics (engineering rules of thumb),
 - process discipline,
 - some capital (e.g. computer aided tools), and
 - man power



AMERICANS



- They see software technology as a vehicle:
 - for creating dedicated software companies
 - that produce at least “good-enough” products and
 - try to set industry standards as well as
 - make lots of money in the process
- Netscape:
 - easy-to-use browser that run on windows as well as other software platforms.
- Bill gates: see a personal computer on every desktop.
- Software programming can still be science for Americans.

BUSINESS OBJECT

- Business intelligent company
- Focus on oracle- oracle sales (France)
- Five years: nothing to \$60 million
- IBM credibility (only 5%)



I2 TECHNOLOGIES

- factory planning, logistics, and SCM
- technical in India
- never really had billion dollar business



The Supply Chain Company™

GOOD TIME VS. BAD TIME

- when time are good,
 - it is easy for software product companies to grow revenue,
 - perhaps to a billion dollars or more within a few short years,
- But when time are bad,
 - revenue can collapse like a stone falling to earth
 - because customers can simply stop buying new products.
- survive:
 - solid based of loyal,
 - satisfied customers,
 - who pay “recurring” fees over long term contracts for products updated, buy fixes, customized, and other services

OUTLINE-1

- Terminology of Software Business
- Software Business Ecosystem
- Software Business Functions
- Software Industry Analysis
- Software Business Strategy

SOFTWARE BUSINESS STRATEGY

- Porter model “five forces”
 - power of buyers,
 - power of suppliers,
 - intensity of rivalries,
 - substitutes, and
 - entry barriers



BASIC QUESTIONS ABOUT PRODUCTS, MARKETS & STRATEGIC POSITIONING

- Want to be product or service company?
- Sell to individuals or enterprises, or mass or niche markets?
- How horizontal (broad) or vertical (specialized) is your product or service?
- Can you generate a recurring revenue stream to endure in good times and bad?
- Target mainstream customers, or plan to avoid “the chasm”?
- Hope to be leader, follower, or complementary?
- What kind of character do you want your company to have?

BASIC QUESTIONS ABOUT PRODUCTS, MARKETS & STRATEGIC POSITIONING

- Want to be product or service company?
- Sell to individuals or enterprises, or mass or niche markets?
- How horizontal (broad) or vertical (specialized) is your product or service?
- Can you generate a recurring revenue stream to endure in good times and bad?
- Target mainstream customers, or plan to avoid “the chasm”?
- Hope to be leader, follower, or complementary?
- What kind of character do you want your company to have?

BASIC QUESTIONS ABOUT PRODUCTS, MARKETS & STRATEGIC POSITIONING

- Want to be product or service company?
- Sell to individuals or enterprises, or mass or niche markets?
- How horizontal (broad) or vertical (specialized) is your product or service?
- Can you generate a recurring revenue stream to endure in good times and bad?
- Target mainstream customers, or plan to avoid “the chasm”?
- Hope to be leader, follower, or complementary?
- What kind of character do you want your company to have?

WANT TO BE PRODUCT OR SERVICE COMPANY?

- Not black and white, but spectrum:
 - Customize software.
 - Microsoft (service + product)
 - Reuse partial products (Ernst & Young) but large revenue from service
 - Product company but awfully close to service
 - E.g. PeopleSoft, SAP
- In bad times, customers (individuals or enterprises)
 - may decide not to buy new versions of the software products they are using.
 - considerable Risk of revenue,
 - for it is discretionary in nature

WANT TO BE PRODUCT OR SERVICE COMPANY?

- Need for balance of product and service revenue
 - to survive in bad times and
 - to grow rapidly and profitably in good times.
- Ability to greatly increase-as much as
 - double or triple- revenue over time through
 - accumulation of contracts for service and custom software,
 - including maintenance even if
 - new software products sales lag behind in growth rates. (large enterprise customer, \$ each license)

WANT TO BE PRODUCT OR SERVICE COMPANY?

- Perpetual license to customer, as long as it pays the agreed-upon annual maintenance fee
 - right to upgrade to new version
- Long term contracts for service and maintenance
 - somewhat like that of a bank
- Business model of software products companies:
 - in contrast, because they can replicate copy after copy with
 - minimal marginal costs,
 - more like that of a printing press.

WANT TO BE PRODUCT OR SERVICE COMPANY?

- Hybrid solution model:
 - sell mixture of products and services,
 - including maintenance upgrades and maintenance of special product.
 - reason: their technologies are often too complex to package as “off the shelf” products.
 - So they sell customized solution, or special integration and installation work)
- It is usually very difficult for enterprise customers to switch from this type of software,
 - which often runs critical functions in the firm.

WANT TO BE PRODUCT OR SERVICE COMPANY?

- Hit software package is like having bestselling book.
 - hard, one hit continuous stream of new product even harder
- Challenge of saturated market:
 - customers already have enough or “good-enough” products.
- Tension that inevitably emerges in enterprise software companies with a strong products business:
 - They know they must eventually move toward selling
 - more labor-intensive and
 - less profitable services.
- Use revenue of service to support their new product sales.

WANT TO BE PRODUCT OR SERVICE COMPANY?

- Service revenue will rise or fall in proportion to
 - the rise or fall in new product sales,
 - unless the company can decouple services from product sales (e.g. SAP, IBM)
- What we see is that service revenues
 - have been growing in both absolute terms and
 - relative to product sales and
 - in some cases have exceeded product sales
- The bottom line is that the distinction between a products company and a services company in the software business is not always clear or desirable.

WANT TO BE PRODUCT OR SERVICE COMPANY?

- In many ways, hybrid solutions companies may have the best business model:
 - They can leverage some core technologies like products companies
 - But also generate recurring revenues like services companies.
 - Over a few years, they can double or triple what their revenues would have been with little or no services.
- Furthermore:
 - If a company can decouple service contracts from product contracts,
 - it has the potential to grow even faster than a more conventional enterprise software company that
 - only sells service and maintenance with its own new products.

WANT TO BE PRODUCT OR SERVICE COMPANY?

- Managers of these kinds of software companies (Hybrid solution model):
 - still need to choose a primary strategic orientation and
 - understand the potential consequences of their decisions.
 - The reason is that selling mainly software products to new customers requires:
 - very different strategies,
 - organizational capabilities, and
 - financial investments
 - compared to selling mainly software services and product upgrades to an existing customer base.

WANT TO BE PRODUCT OR SERVICE COMPANY?

- Volume sales: basic growth strategies:
 - scaling or duplicating. (standards by Microsoft)
 - The development organization needs to focus on:
 - Creating a stream of new products and upgrades
 - that appear at regular intervals
 - with standardized features that are “good enough” for the largest possible set of users.
 - Mass marketing and distribution skills are critical
 - Part of the strategy for a products company might also include:
 - trying to become a platform leader

WANT TO BE PRODUCT OR SERVICE COMPANY?

- The software services business:
 - Including the services side of the business in a products company
 - is mainly about people and building specific (not general) customer relationships.
 - It is about getting enough profitable accounts to keep your consultants and developers busy close to 100 percent of the time. (IBM, Ernst & Young)
- These types of services are not scale businesses.
- But services companies can be strategic as well as efficient.

WANT TO BE PRODUCT OR SERVICE COMPANY?

- The software services business
 - It is usually important to mix:
 - Senior people with junior to maximize profits for any given client project,
 - Although the challenge is for a services firm to do this without damaging
 - The relationship by having inadequately skilled people on the job.
- For software products companies, again, the lure is:
 - potentially enormous *economies of scale*
 - that come from selling multiple copies of the same piece of software.
- For software service companies:
 - *economies of scope* are the Holy Grail **to** strive for, and
 - these are more elusive.

WANT TO BE PRODUCT OR SERVICE COMPANY?

- Economy of Scope, can come from:
 - structuring knowledge such as:
 - how to do requirements,
 - manage projects,
 - customize applications,
 - conduct user acceptance testing,
 - or reuse design frameworks and
 - even pieces of code across different projects and customers
- Economies of scope can also come from:
 - clever account management
 - forming relationships with particular customers
 - who buy a lot of your software and services over time.

WANT TO BE PRODUCT OR SERVICE COMPANY?

- It is a crude rule of thumb:
 - companies can have high sales productivity and still lose lots of money by overspending in R and D, sales and marketing, or general administration
- certain boundaries of expenses that should be kept :
 - For sales and marketing, this should be about 25 to 30 percent of total revenues;
 - Research and development about 10 to 15 percent;
 - General administration about 5 percent.
 - A fair profit target is 20 to 30 percent of revenues as operating income.

WANT TO BE PRODUCT OR SERVICE COMPANY?

- For many software products companies:
 - services such as customization, installation, and integration support are necessary to
 - drive new-product sales.
- Service expenses consist of:
 - expenses related to technical support,
 - consulting, training, and other services.
- Cost of software license fees consists:
 - mainly of materials such as compact discs and printed manuals,
 - packaging, freight, inventory, third-party royalties, and
 - amortization expense related to capitalized software development costs.

WANT TO BE PRODUCT OR SERVICE COMPANY?

- Gross margins of are of limited value as a metric:
 - High R&D Cost
 - High sales and marketing expenses (went toward selling both products and services, because of high head count)
 - Head count is relatively high because their revenue are relatively low—an economies-of-scale problem.
- Investors such as venture capitalists generally:
 - have a strong preference for software products companies:
 - because they have a much greater potential for rapid growth and profits.
- Hybrid firms that try to take over:
 - more of this services business themselves
 - will create some conflicts with their “channel partners.”

BASIC QUESTIONS ABOUT PRODUCTS, MARKETS & STRATEGIC POSITIONING

- Want to be product or service company?
- Sell to individuals or enterprises, or mass or niche markets?
- How horizontal (broad) or vertical (specialized) is your product or service?
- Can you generate a recurring revenue stream to endure in good times and bad?
- Target mainstream customers, or plan to avoid “the chasm”?
- Hope to be leader, follower, or complementary?
- What kind of character do you want your company to have?

SELL TO INDIVIDUALS OR ENTERPRISES, OR MASS OR NICHE MARKETS?

- In general, though, selling to medium-sized and large enterprises requires:
 - very different business models and organization capabilities compared to selling to individuals.
- Should you tackle the mass market,
 - where the payoffs can be enormous but the competition may be stiff,
- Or select particular niches
 - where you have a greater chance of establishing a distinct advantage?
- As in other industries:
 - the average profitability levels in these different segments vary.

SELL TO INDIVIDUALS OR ENTERPRISES, OR MASS OR NICHE MARKETS?

- Clearly, some segments:
 - have less competition than others and
 - companies can charge higher prices or
 - more easily benefit from economies of scale to lower their costs per unit sold.
- Some segments, such as services, can be very labor-intensive.
- Some mass markets:
 - such as those for operating systems, database products, and shrink-wrapped applications,
 - can achieve tremendous economies of scale since replicating the products costs so little.

SELL TO INDIVIDUALS OR ENTERPRISES, OR MASS OR NICHE MARKETS?

- Enterprise software companies have to:
 - offer “whole product solution” in order
 - to cross successfully into the mainstream market.
 - This means that the software product must be:
 - rock-solid in terms of reliability and
 - include easy-to-use features for late adopters, good documentation,
 - thorough technical support, and a full array of complementary products and services.

SELL TO INDIVIDUALS OR ENTERPRISES, OR MASS OR NICHE MARKETS?

- Software companies that sell to enterprises:
 - generally offer a combination of
 - standardized products sold through software license fees
 - and services with maintenance sold through separate multiyear contracts.
- Software companies that target individual:
 - consumers or small enterprises for the most part
 - sell standardized products
 - The latter sell either directly to individuals or to intermediaries.

SELL TO INDIVIDUALS OR ENTERPRISES, OR MASS OR NICHE MARKETS?

- Enterprises may produce very high revenues per sale:
 - high “average selling prices”
 - but these revenues can be costly.
 - expectation 30% license, 70% service and maintenance)

SELL TO INDIVIDUALS OR ENTERPRISES, OR MASS OR NICHE MARKETS?

- Negative Side (Enterprise):
 - Needs large direct sales force, more costly than consumer sales
 - changed with Web, but not completely
 - Customized feature need
 - Product development has to proceed more or less on schedule,
 - even with the distraction of custom work
 - Give large discount to close a sale
 - end of a quarter, or fiscal year
 - Like Book publisher, discretionary nature, problem sales, if previous version is still “good enough”

SELL TO INDIVIDUALS OR ENTERPRISES, OR MASS OR NICHE MARKETS?

○ Positive Side(Enterprise):

- Despite the potentially high cost of sales, when a software company targets enterprises, it has more ways of succeeding, even if it fails to develop a best-seller product.
- Being able to tailor products to meet needs of particular customer or type of customer
 - charge for custom features.

○ Positive Side(Enterprise):

- Offer training and consulting to help use its products, and other more effectively.
- Clever maintenance and upgrade schedule
 - generate continuing revenue stream, or perpetual licenses

SELL TO INDIVIDUALS OR ENTERPRISES, OR MASS OR NICHE MARKETS?

- Enterprise customers are usually willing to pay for:
 - Detailed product documentation,
 - Expert technical support,
 - Training, and
 - Specialized integration work to link one product with other products or databases.

SELL TO INDIVIDUALS OR ENTERPRISES, OR MASS OR NICHE MARKETS?

- The software business, as in other businesses:
 - There are mass-market companies as well as niche-oriented companies:
 - Database (Oracle, IBM, Informix, Microsoft SQL Server, MS Access)
 - Operating Systems (Windows 200 and XP, Unix (Sun, IBM, HLP),, Apple (niche, easy-to-use computer, graphic work), Linux)
- As in other industries, selling effectively to different types of customers usually requires:
 - very different strategies and organizational capabilities
 - Less competition more profitable, dominant player or platform leader: Microsoft

BASIC QUESTIONS ABOUT PRODUCTS, MARKETS & STRATEGIC POSITIONING

- Want to be product or service company?
- Sell to individuals or enterprises, or mass or niche markets?
- How horizontal (broad) or vertical (specialized) is your product or service?
- Can you generate a recurring revenue stream to endure in good times and bad?
- Target mainstream customers, or plan to avoid “the chasm”?
- Hope to be leader, follower, or complementary?
- What kind of character do you want your company to have?

HOW HORIZONTAL (BROAD) OR VERTICAL (SPECIALIZED) IS YOUR PRODUCT OR SERVICE?

- Product Lines & Market Segmentation
- Combination is possible by:
 - Developing general-purpose products
 - Specialized for or tailored to particular industries or types of users.
- Must also decide individual products or product suites?
- By horizontal, mean:
 - A potential market that covers most or all PC users,
 - Regardless of their industry or functional specialization.
 - (e.g. Microsoft)
 - more likely to be mass markets, much greater potential sales.
-

HOW HORIZONTAL (BROAD) OR VERTICAL (SPECIALIZED) IS YOUR PRODUCT OR SERVICE?

- By vertical, mean:
 - a market that lies in a specific domain.
- Domain may consist of:
 - Industry
 - e.g. healthcare
 - Technical specialty
 - e.g. computer aided design
 - Technical specialty for a particular industry
 - e.g. combination of previous
 - Platform specific market (particular platform).

HOW HORIZONTAL (BROAD) OR VERTICAL (SPECIALIZED) IS YOUR PRODUCT OR SERVICE?

- Segmenting markets both horizontally and vertically can provide:
 - software companies (and other companies)
 - with a blueprint for diversification and expansion
 - while still remaining close to their core of expertise.
- Vertical markets that are exactly right for your solution must be easier to conquer than horizontal markets that require general-purpose solutions that anybody can use.

HOW HORIZONTAL (BROAD) OR VERTICAL (SPECIALIZED) IS YOUR PRODUCT OR SERVICE?

○ Lure of Horizontal:

- Different OS, H/W platform:
 - When this occurs in a firm with multiple products, it will have to subsidize weak offerings
 - costs for design, engineering, testing, documentation, maintenance, and support on the technical side, as well as costs for sales and marketing
 - But not all are equally profitable or large in terms of users.
- A more serious dilemma is that horizontal markets can require:
 - enormous investment and skill to master.
 - shouldn't play at least while having lack the resources
- It would have been great to become a major horizontal player,
 - but first the company should have created a few compelling pilot applications as a proof of concept for the technology.

HOW HORIZONTAL (BROAD) OR VERTICAL (SPECIALIZED) IS YOUR PRODUCT OR SERVICE?

○ Lure of Product Bundling: (MS Office)

- Software companies also have a choice of selling their products one at a time or
 - bundling them into multiproduct “suites” or “solutions.”
 - idea: broader or more horizontal spectrum of customers- more mass market, with lower prices: for economy of scale
- The danger is that a software company may bundle weak products with strong ones
 - and not improve them. (huge investigation needed),
 - The resulting “suite” then does little more than subsidize
 - weak engineering or sales and marketing groups and raise costs while lowering or eliminating profits.

HOW HORIZONTAL (BROAD) OR VERTICAL (SPECIALIZED) IS YOUR PRODUCT OR SERVICE?

○ Lure of Product Bundling: (MS Office)

- Some bottom lines here: Every software company should periodically reevaluate its products to determine if it is getting the breadth of offerings right.
- Some products seem as though they could be horizontal blockbusters,
 - but they may need a vertical foothold to start.

HOW HORIZONTAL (BROAD) OR VERTICAL (SPECIALIZED) IS YOUR PRODUCT OR SERVICE?

- Lure of Too much Growth and Diversification:
 - Various data from different industries support the observation that :
 - rapidly growing companies have much better chances of survival.
 - Three basic strategies for rapid growth:
 - Scaling:
 - doing more of what you are already doing,
 - such as launching a bigger marketing and sales campaign
 - to sell more of your software to similar customers
 - E.g. i2, Business Object, Netscape
 - Duplicating:
 - primarily extending your same strategy to other geographical markets
 - e.g., going overseas
 - Granulating:
 - diversifying product offering and the organization structure by
 - creating new, small business units to target new product opportunities
 - E.g. SAP: new biz unit for advantage

HOW HORIZONTAL (BROAD) OR VERTICAL (SPECIALIZED) IS YOUR PRODUCT OR SERVICE?

- Lure of Too much Growth and Diversification:
 - Diversification into related product lines and technologies is usually a good way to grow, compared to unrelated diversification.
 - Declining hardware sales meant declining demand for software applications. (Infinium Software: spend on NT acquisition)
 - Not even long-standing customers seemingly locked into your software will remain loyal if they believe that the vendor has stopped investing in the products and is not willing to help them incorporate new technologies, such as interoperability with the Web.
 - He was wrong about how easy it would be to diversify by acquiring companies building applications for the NT platform. Two thirds of all acquisitions fail, historically. infinium's acquisitions record was not even this good.

HOW HORIZONTAL (BROAD) OR VERTICAL (SPECIALIZED) IS YOUR PRODUCT OR SERVICE?

- Diversification into related product lines and technologies is usually a good way to grow,
 - compared to unrelated diversification.
- Declining hardware sales meant declining demand for software applications.
 - E.g. Infinium Software: spend on NT acquisition
- Not even long-standing customers seemingly locked into your software will remain loyal
 - if they believe that the vendor has stopped investing in the products and
 - is not willing to help them incorporate new technologies,
 - such as interoperability with the Web.
- He was wrong about:
 - how easy it would be to diversify by acquiring companies building applications for the NT platform.
 - Two thirds of all acquisitions fail, historically. Infinium's acquisitions record was not even this good.

BASIC QUESTIONS ABOUT PRODUCTS, MARKETS & STRATEGIC POSITIONING

- Want to be product or service company?
- Sell to individuals or enterprises, or mass or niche markets?
- How horizontal (broad) or vertical (specialized) is your product or service?
- Can you generate a recurring revenue stream to endure in good times and bad?
- Target mainstream customers, or plan to avoid “the chasm”?
- Hope to be leader, follower, or complementary?
- What kind of character do you want your company to have?

CAN YOU GENERATE A RECURRING REVENUE STREAM TO ENDURE IN GOOD TIMES AND BAD?

- It is hard for enterprise customers to switch:
 - when their data and procedures are deeply embedded into a particular vendor's software,
 - usually through custom code or special configurations of the applications.
- Services and maintenance agreements for product upgrades and enhancements are important :
 - because the goal of every software company should be to establish large and growing revenues that are recurring

CAN YOU GENERATE A RECURRING REVENUE STREAM TO ENDURE IN GOOD TIMES AND BAD?

- The best way is long-term contracts for maintenance
 - such as product upgrades, tailored enhancements, and regular bug fixes
- And a variety of services
 - such as IT consulting, system integration, training, and technical support
- For most enterprise software companies:
 - product sales are the engine
 - that drives that recurring stream of service and maintenance revenues.

CAN YOU GENERATE A RECURRING REVENUE STREAM TO ENDURE IN GOOD TIMES AND BAD?

- Even software companies that do not have long-term contracts
- to generate future income can achieve predictable revenue streams.
 - Automobile: persuade customer to “upgrade” to new model.
 - HP & Canon:
 - predictable recurring revenue stream from the sales of an essential (and costly) complement-toner cartridge
 - Microsoft:
 - recurring revenue from both platform products and complement:
 - work closely with hardware complementary, such as Intel, PC manufacturer, Dell and Compaq.
 - Force customer and enterprise to upgrade their software:
 - incompatibility with old version, grantee backward compatibility,
 - but not forward compatibility,
 - Lack of compatibility forces upgrade that companies can often predict.

CAN YOU GENERATE A RECURRING REVENUE STREAM TO ENDURE IN GOOD TIMES AND BAD?

- Blurring the distinction between product and services:
 - Renting software rather than buying it:
 - Application Service Provider (ASP) as well as some uses of Web Services.
 - Long term maintenance or technical support contracts,
 - with right to receive future product upgrades.
 - Called subscription model.
 - Product sales complicates accounting, in that:
 - companies must accumulate what are called “deferred revenue”,
 - can recognize these revenues only as they deliver the software upgrades in the future,
 - and creates some opportunities for abuse.

BASIC QUESTIONS ABOUT PRODUCTS, MARKETS & STRATEGIC POSITIONING

- Want to be product or service company?
- Sell to individuals or enterprises, or mass or niche markets?
- How horizontal (broad) or vertical (specialized) is your product or service?
- Can you generate a recurring revenue stream to endure in good times and bad?
- Target mainstream customers, or plan to avoid “the chasm”?
- Hope to be leader, follower, or complementary?
- What kind of character do you want your company to have?

TARGET MAINSTREAM CUSTOMERS, OR PLAN TO AVOID “THE CHASM”?

- Challenge of how to sell conservative mainstream customers rather than leading-edge users.
- Grow of sale continuously across the different market segments
- Chasm between Early Adopters and Early Majority
- Early Adopters:
 - buying change agents, trying to get a jump on the competition
 - prepared to deal with radical changes, discontinuities and bugs.
- By contrast, the Early Majority:
 - want productivity, improvement tools for existing operations.
 - They don't want to debug somebody else's new product.

TARGET MAINSTREAM CUSTOMERS, OR PLAN TO AVOID “THE CHASM”?

- As a result:

- The Early Adopters do not make good references for the Early Majority.
- But the Early Majority will not buy without good references

- Solution:

- Moore advises companies to cross the chasm by focusing on:
 - one or two market niches
 - where they can afford to develop “whole product solution”
 - that provide the service and support
 - mainstream market customers (conservatives) want.

TARGET MAINSTREAM CUSTOMERS, OR PLAN TO AVOID “THE CHASM”?

- To get to the mainstream, companies must also
 - cultivate reference customers and relationships with these customers.
- The niche should be a strategic “target market”
 - in the sense that it can lead to a larger set of customers.

TARGET MAINSTREAM CUSTOMERS, OR PLAN TO AVOID “THE CHASM”?

- Also, Moore argued that:
 - once in the mainstream,
 - companies no longer have to be state of the art,
 - but just “good enough,” although they need to stay close to the technology leaders.
- Furthermore, to develop the mainstream market:
 - Companies must shift their marketing from “product-centric”
 - fastest product, easiest of use, most elegant architecture, cheapest price, unique functionality
 - To “market-centric”
 - largest installed base, most third-party supporters, de facto standard, lowest cost of ownership, best quality support

TARGET MAINSTREAM CUSTOMERS, OR PLAN TO AVOID “THE CHASM”?

- Biggest Message from Moore's book(Cross chasm model):
 - Dilemma: A log of new companies misread their early market success,
 - invest heavily in sales and distribution net,
 - saturation of Early Adopter,
 - stagnation of sales for un easiness,
 - not enough scalability,
 - R&D gets bogged down dealing with mainstream customer problems or
 - special projects,
 - rather than continue innovation.
 - Fall off a cliff of companies sales.

TARGET MAINSTREAM CUSTOMERS, OR PLAN TO AVOID “THE CHASM”?

- Competing in niche markets is very different from competing in mass markets
 - And managers may not learn from niche marketing,
 - what they need to know to become a truly successful mass-market player.
- Apple failed to expand the Mac's base into the mainstream market for reasons other than the chasm- crossing problem:
 - Apple kept prices high,
 - didn't license early,
 - didn't get enough machines into the marketplace to attract as many complements (software) as the competition,
 - and didn't keep innovating in the software, which allowed Microsoft to catch up.

TARGET MAINSTREAM CUSTOMERS, OR PLAN TO AVOID “THE CHASM”?

- Managers should understand dynamic of mass-market standard,
 - how standards emerge in their industries
 - to help them cross the chasm and succeed in the mainstream market.
- Simply crossing the chasm is no guarantee of long-term success.

TARGET MAINSTREAM CUSTOMERS, OR PLAN TO AVOID “THE CHASM”?

- There is the need to control seven critical assets:
 - an installed base,
 - intellectual property rights,
 - ability to innovate,
 - first-mover advantage,
 - manufacturing abilities,
 - complementary producers, and
 - brand/reputation.

TARGET MAINSTREAM CUSTOMERS, OR PLAN TO AVOID “THE CHASM”?

- Ways to accelerate the process of getting into a mainstream market from niche beginning or a zero start:
 - Credibility built with customer in one market
 - to help them enter new markets
 - without having to go through all the steps of Moore.
 - E.g. IBM, Microsoft Office, SQL Server
 - Company can be “handled” the mass-market standard
 - and use it to leapfrog the chasm
 - E.g. DOS and complementary assets, spreadsheet, word processor, buying ...

BASIC QUESTIONS ABOUT PRODUCTS, MARKETS & STRATEGIC POSITIONING

- Want to be product or service company?
- Sell to individuals or enterprises, or mass or niche markets?
- How horizontal (broad) or vertical (specialized) is your product or service?
- Can you generate a recurring revenue stream to endure in good times and bad?
- Target mainstream customers, or plan to avoid “the chasm”?
- Hope to be leader, follower, or complementary?
- What kind of character do you want your company to have?

HOPE TO BE LEADER, FOLLOWER, OR COMPLEMENTARY?

- In many high-tech markets, the key issue is not how advanced your technology is
 - and whether you are first to market or not,
 - but whether you are in a position to become the market leader.
- You might also have a core technology that other companies
 - can build complementary products and services around,
 - which presents the opportunity of becoming a platform leader as well as the market leader.

HOPE TO BE LEADER, FOLLOWER, OR COMPLEMENTARY?

- Complementary products
 - such as PC hardware peripherals or software applications
 - are often essential to make software and hardware platforms
 - such as the Intel-Windows PC, successful
- Sixth industry force is power of complementary in Michael Porter's set of five forces that determine level of industry structure and profitability.

HOPE TO BE LEADER, FOLLOWER, OR COMPLEMENTARY?

- Technology Leaders are not necessarily Market leaders:
 - Xerox: Canon,
 - Ampex (VCR): Sony, Matsushita
 - Xerox spreadsheet, GUI: Microsoft & Apple
 - WordStar of MicroPro: WordPerfect: MS Word
 - VisiCalc commercial Spreadsheet: Lotus1-2-3: MS Excel
 - Netscape: Microsoft
- It should be obvious that technology pioneers
 - are like other pioneers in the old American West:
 - they often make strategic or technical mistakes
 - and get the business equivalent of arrows in the back.

HOPE TO BE LEADER, FOLLOWER, OR COMPLEMENTARY?

- Being first with an invention
 - or the commercialization of a new technology
 - is no guarantee of long-term financial success.
- Microsoft:
 - has succeeded by copying the ideas of others,
 - improving products incrementally and relentlessly,
 - and then overwhelming the market with volume and low prices.
 - Microsoft's products usually come close enough by version three to the market leaders
 - in features and quality to compete effectively with them.
 - Cut price, large amount
- Software companies that are in a new market early have
 - a good chance for their products to become standards or
 - platforms, which can eventually make it difficult for customers to switch from one vendor to another.

HOPE TO BE LEADER, FOLLOWER, OR COMPLEMENTARY?

- Successful leaders:

- Intuit: TurboTax, QuickBook
- Cognos, Business Object: intelligent products
- Sible: CRM
- Netscape (then AOL): Internet software.



HOPE TO BE LEADER, FOLLOWER, OR COMPLEMENTARY?

- Not all industries have platforms that become dominant.
 - But many high-tech mass markets and niche markets do gravitate toward a standard
 - because they depend on core products and complementary products that are compatible.
- To be platform leader
 - company does not have to be first in market or have the best technology
 - E.g. MS DOS
- Platform leaders need to drive innovation around
 - their particular platforms at the broad industry level
 - because the value of the platform usually increases
 - when there are more complements available
 - E.g. work closely with other firms
 - called “network externalities” and “bandwagon” or “positive feedback”

HOPE TO BE LEADER, FOLLOWER, OR COMPLEMENTARY?

- Four “levers” in Platform Leadership:
 - Scope of the firm:
 - what complement platform producer makes itself
 - versus what it encourages or allow other firms to make.
 - Product technology
 - architecture, interfaces, and intellectual property
 - need to decide on the degree of modularity
 - for their product architectures and
 - the degree of openness of the interfaces to the platform.

HOPE TO BE LEADER, FOLLOWER, OR COMPLEMENTARY?

- Four “levers” in Platform Leadership:
 - Relationships with external Complementors
 - need to determine how collaborative or
 - competitive they want the relationship
 - between themselves and their complementors to be.
 - E.g. creating consensus, resolve conflicts
 - Internal organization
 - can reorganize to deal with both external and internal conflicts of interest.
 - Chinese wall between core and complementary: Microsoft and Intel

HOPE TO BE LEADER, FOLLOWER, OR COMPLEMENTARY?

- The reality is that most software companies
 - are not platform leaders;
 - they are complementors of somebody else's platform.
 - more risky strategic decisions
- Complementors always run the risk that
 - they may fail to keep up with changes in their target platform.
 - Or the platform leader may decide to absorb their products into the platform
 - and make the complementary products itself.
- Challenging platform leaders can also be dangerous for complementors.

BASIC QUESTIONS ABOUT PRODUCTS, MARKETS & STRATEGIC POSITIONING

- Want to be product or service company?
- Sell to individuals or enterprises, or mass or niche markets?
- How horizontal (broad) or vertical (specialized) is your product or service?
- Can you generate a recurring revenue stream to endure in good times and bad?
- Target mainstream customers, or plan to avoid “the chasm”?
- Hope to be leader, follower, or complementary?
- What kind of character do you want your company to have?

WHAT KIND OF CHARACTER DO YOU WANT YOUR COMPANY TO HAVE?

- Strategy should also involve determining what kind of an organization you want to be.
- Suitable point of aggressiveness
 - antitrust, exchange commission, customer ,...
- Individual customers, billion-dollar corporations, and trillion-dollar governments put their trust in computer systems
 - necessity of uncompromising ethics
- Too many software companies and other high-tech firms,
 - at one time or another, find themselves guilty
 - or almost guilty of recognizing revenues they should not.

WHAT KIND OF CHARACTER DO YOU WANT YOUR COMPANY TO HAVE?

- One problem, even among companies that do not intentionally try to break the rules,
 - is that the rules are subject to interpretation—
- Aggressive or conservative interpretation and everything in between.
 - Beta software are not “finished” according to rules
 - When it still needs a lot of work to complete features or fix defect defined it is delivered
 - Recognize revenue and then large postship costs to finish product
 - Demand refund or sue the vendors in court
 - Mislead people by recognizing revenue from maintenance and upgrade contracts, different time.
 - E.g. Baan ERP: To bolster sales,
 - top management began playing a virtual shell game with “sales” to distributors that were really subsidiaries it owned.
 - Bad acquisitions, stock price and customer orders collapse after they understand this

WHAT KIND OF CHARACTER DO YOU WANT YOUR COMPANY TO HAVE?

- Losing customer confidence can start by losing credibility.
- *Like its close Cousin, integrity, credibility is something that most firms, or individuals, can lose only once.*
- Well-managed companies try their best to make good on their promises,
 - even if they lose money in the process
 - build discipline into its sales agreement and product specifications, as well as add buffer time to schedule and protect its debugging and testing time
- Poorly managed companies try to “get the business”
 - at almost any cost and cut corners on what they deliver or
 - deliver late because they know they overpromised to make the deal.

OUTLINE-2

- Software Business Operations
- Software Business Entrepreneurship
- Open Source Software Initiation
- Open Source Business Models and Revenue
- Open Source Software Operations
- Commercial Open Source

HISTORY: IBM SURVIVAL

- *First: Gerstner decided to make services in a neutral sense*
 - *as important as or perhaps more important than the company's proprietary hardware and software products.*
- Second, Gerstner decided to embrace "open systems"
 - that is, he would make IBM adopt standard protocols
 - so that IBM software could run on different types of hardware,
 - including competitors' machines, and IBM hardware could run different types of software. (Java, Apache,...)
- Third, in 1995, Gerstner decided to
 - embrace the Internet and meld it into a vision of "networked computing."
 - suitable for mainframes, WebSphere,
 - and popular "middleware" for linking different applications and processing transactions
 - In addition, the consulting arm of the company quickly learned how to help clients conduct "e-business."

LESSONS FROM HISTORY

- First for start-ups and established firms in the enterprise software field,
 - *the key choice is not simply whether to be a services company or a products company,*
 - *but how much emphasis to place on one type of business over the other.*
- Second, history suggests that *which part of the business to emphasize more than the other*
 - *should change at different times in the evolution of a software company's customer base and product lines.*
- A third point is that *changes in platform technologies*
 - *generate new demand-both for new products and for new services*
 - *to help customers reuse their existing software assets.*

LESSONS FROM HISTORY

- Hardware companies that offer total solutions, including hardware, software, and services
 - E.g. IBM, DEC, and Sun
 - are rarely able to cover all horizontal and vertical software segments,
 - and sometimes they do not have the skills to build systems software for a new platform.
- Fourth, the history of the business suggests that:
 - *niche applications and new platforms*
 - *are the best places to look for new software product and service opportunities.*

NATO REPORT ON SOFTWARE ENGINEERING PROBLEMS (1968)



- lack of understanding in system requirements on the part of customers and designers.
- large gaps between estimates of costs and time with actual expenditures due to poor estimating techniques,
 - failure to allow time for changes in requirements,
 - and division of programming tasks into blocks
 - before the divisions of the system are understood well enough to do so properly.
- Large variations, as much as 26: 1 in one study, in programmers' productivity levels.
- Difficulty of dividing labor between design and production (coding),
 - since design-type decisions must still be made during coding.
- Difficulty in monitoring progress in a software project,
 - since "program construction is not always a simple progression in which each act of assembly represents a distinct forward step."

NATO REPORT ON SOFTWARE ENGINEERING PROBLEMS (1968)



- Rapid growth in size of software systems. Poor communication among groups working on the same project,
 - exacerbated by too much uncoordinated or unnecessary information and
 - a lack of automation to handle necessary information.
- Large expense of developing online production control tools.
- Difficulty of measuring key aspects of programmer and system performance.
- A tradition among software developers of not writing systems "for practical use"
 - but trying to write new and better systems,
 - so that they are always combining research, development, and production in a single project,
 - which then makes it difficult to predict and manage.
- Rapid growth in the need for programmers and insufficient numbers of adequately trained and skilled programmers.

NATO REPORT ON SOFTWARE ENGINEERING PROBLEMS (1968)



- Difficulty of achieving sufficient reliability (reduced errors and error tolerance) in large software systems.
- Dependence of software on hardware,
 - which makes standardization of software difficult across different machines.
- Lack of inventories of reusable software components to aid in the building of new programs.
- Software maintenance costs often exceeding the cost of the original system development. .

PERSISTENT OF SIMILAR PROBLEM

- The persistence of similar problems over decades:
 - as well as observations that as many as 75 to 80 percent of software projects are typically late
 - and over budget, suggests that the field of software development has *not* made enough progress.

CHALLENGES

- Writing program algorithms is usually not a routine activity.
- It generally involves creativity and invention on some level,
 - as well as problem solving and trial and error
- In custom software projects, users often do not know what they want until they see part of the system in front of them.
- In the products market, users often want compelling features before they will make a purchase.

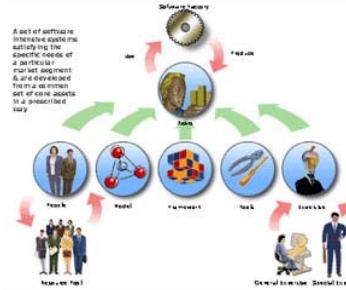
CHALLENGES

- But exacerbating the difficulties is the fact that:
 - many software development organizations too often seem to want to reinvent the wheel
 - when it comes to managing projects and thinking about process.
- Different requirements of projects:
 - mission critical,
 - invention and innovation which are difficult to structure and predict
 - treating as unique events

SOFTWARE FACTORIES

- To make software development more tractable, managers at companies dreamed of creating “software factories”
 - E.g. Microsoft, AT&T, Japanese
- Software development is not a manufacturing activity;
- it is more a form of product design,
 - where the design is the product and replication is trivial.
 - The design process, moreover, has some unique characteristics,
 - requiring a combination of art, science, engineering, and management skills,
 - especially in new applications and large projects.
 - These characteristics make the design, construction, testing, and maintenance of software systems somewhat difficult to control.

SOFTWARE FACTORIES



- Yet it is also true that some companies have created:
 - *factory-like* organizations and processes
 - to develop certain kinds of software and
 - to manage certain kinds of projects with particular kinds of people
- Some factory organizations have tried to separate the requirements generation and high-level design process from:
 - the program construction (coding and debugging)
 - have outsourced programming and testing work to subsidiaries or overseas contractors (such as in India).

SOFTWARE FACTORIES

- Other organizations have had less success or have failed for a variety of reasons, including:
 - the difficulty of managing iterations in development,
 - understanding changing customer requirements,
 - standardizing development methods for widely varying contexts, and
 - reusing code systematically.



SOFTWARE FACTORIES

- Fast-paced markets with rapidly changing technologies and customer requirements:
 - such as PC and Internet software in their early years,
 - seem poorly suited to structured methods of software development.
- Talented, creative programmers also usually
 - do not fit well into factory-like environments.
- mainframes, more suitable



SOFTWARE FACTORIES



- These organizations became highly adept at:
 - Building large-scale industrial software systems
 - that had a great deal of commonality from one project to another.
- Japanese projects reach zero defect, sequential “waterfall”
- At the time, Japan:
 - Did not have a large pool of people trained in computer science to draw on.
 - Accordingly, standardized development methods,
 - reuse libraries
 - computer-aided tools, and
 - extensive in-house training served their needs relatively well.

SOFTWARE FACTORIES



- Embedded software development for
 - printers,
 - machine tools,
 - consumer electronics devices, and
 - similar products
- often can use a factory-like waterfall approach effectively and
- even deploy computer-aided tools to generate a lot of the final code automatically

SOFTWARE ENGINEERING INSTITUTE

- *Most* of the practices SEI (Software Engineering Institute, Carnegie Melon University)
 - has recommended derive from IBM's experiences
 - in developing operating systems and
 - complex industrial and government applications,
 - particularly as related by former IBM manager Watts Humphrey.
 - Five level model of Capability Maturity Model (CMM)
- Firms can gain significant savings by following the SEI recommendations.
- The SEI approach seemed to apply best to
 - large firms building complex enterprise software
 - for relatively stable markets

RECENT SOFTWARE ASSESSMENT

○ Project management

- scheduling and project control, planning and resource allocation (of both time and people), and
- risk management measures,
- lot of small schedule but no one in charge of coordinating work
- monitoring a master schedule,
- hard to tell how much work had progressed, under pressured schedule,
- misallocation of people became more serious,
- unplanned changes in designs,
- too much haphazard response to changes,
- not paying enough attention to quality and rushing,
- absence of limit on project scope and deadlines,
- keep making changes in response to demand

RECENT SOFTWARE ASSESSMENT

○ Project management

- Products that shared key components did not have synchronized schedules or independent components,
 - so that one team was continually waiting for another.
- The lack of risk management measures was a problem in that
 - the company had no buffer time in the schedule and no "Plan B" in case the project ran into trouble, which it did.
- The company also did no exploration of technical feasibility for what they wanted to do in the project-and
 - they ended up taking on a challenge akin to "rocket science"
 - without knowing it beforehand.
 - also duplication of work in design, coding, testing, and documentation, for different operating systems

RECENT SOFTWARE ASSESSMENT

○ Design and Development challenge:

- Lack of a comprehensive and effective development strategy and process
- Product design difficult to stabilize, modify, evolve, and migrate
- Chaotic product concept and architectural design process
- Undisciplined functional design and specification process

RECENT SOFTWARE ASSESSMENT

○ Design and Development challenge:

- Little parallel work or automation in development, integration, and testing
- Weak formal and informal reviews *of* designs, code, and documentation
- Poor change control system
- Inadequate customer feedback and design input mechanisms

RECENT SOFTWARE ASSESSMENT

- Design and Development challenge:
 - The main modules in the product suite are technically very tightly coupled and not really separate products.
 - The lack of an interface layer to isolate the product code from target operating system platforms
 - makes the product difficult to migrate to different platforms and
 - difficult to evolve as vendors such as Microsoft modify their APIs.
 - The multiplatform objectives are, therefore, technically correct

RECENT SOFTWARE ASSESSMENT

- Design and Development challenge:
 - Senior managers, consequently, must make a *strategic* decision whether or not to continue pursuing the multiplatform objectives or focus on Windows.
 - The product architecture does not adequately identify common or kernel components
 - i.e., those shared by multiple groups
 - and minimize the interdependencies among them, which makes it difficult to build stable common components.
 - The common components at present need to be too large to contain the shared functionality,
 - making them especially difficult to design, stabilize, evolve, and migrate.

RECENT SOFTWARE ASSESSMENT

- Testing and Quality Assurance:
 - Code file check-in and check-out procedures are not standard-sized,
 - so developers often waste time figuring out what to do and when, or work with outdated files.
 - The process from file check-in to acceptance and global compilation takes one to two days;
 - there is not enough computing power to speed this up.
 - There is no set time on a given day for integration builds

RECENT SOFTWARE ASSESSMENT

○ Knowledge and People Management:

- Architectural and design knowledge vague and not widely shared
- Available product and project information difficult to locate
- Poor communication, feedback, and learning across groups
- Serious overwork and low morale
- Confusing recruitment and assignments
- No training or skills development
- No career path or personnel management system

CHALLENGES

- *Software development groups and projects within the same organization usually need to define*
 - *different kinds of processes*
 - *for different kinds of products, markets, and customer requirements.*
- *Different needs of mission critical softwares and others,*
 - *necessity of having group reviewing process after the project finished.*
- Most important is that managers need
 - a strategy to manage innovation and design,
 - rather than leaving too much to chance and
 - to the developers themselves.

BEST PRACTICE

- Synch-and-stabilize techniques are particularly useful for risk management
 - in the sense that they provide a mechanism for assessing progress in a project and
 - making adjustments continuously.
- The other important idea here is that *late design changes can be good*.
- A process that expects and allows projects to accommodate lots of change
 - with a minimal impact on quality and
 - productivity is
 - a great competitive ad-vantage for many software firms.
- *Economy of Scope*:
 - The idea is to leverage work and creativity across multiple projects, rather than treating each project as a unique undertaking done from scratch.

BEST PRACTICE

- With prioritization and modularization,
 - a team usually has the option to cut lower-priority features if a project falls behind schedule
- It is also possible to evolve the architecture of a software product incrementally
 - to make it more modular
- A firm should devote about 20 percent of its engineering staff to architectural work
 - for ongoing projects.
- Another piece of common wisdom in the software engineering field is
 - that a small team of great people works much better
 - than a large team of mediocre people and that talent is more important than experience.

BEST PRACTICE

- *Super programmers are hard to find and harder to keep.*
- *So the more common problem many software companies have is:*
 - *how to get relatively large groups of people with varying abilities and skills*
 - *to work together like nimble and talented small teams.*

WATERFALL APPROACH

- The most important thing to remember is that:
 - the traditional waterfall model for project management,
 - though it may deliver software on time,
 - matching requirements exactly and with few bugs,
 - is not a good response to fast-paced markets driven by the need to adapt to continuous innovation, uncertainty in customer requirements, and competition.



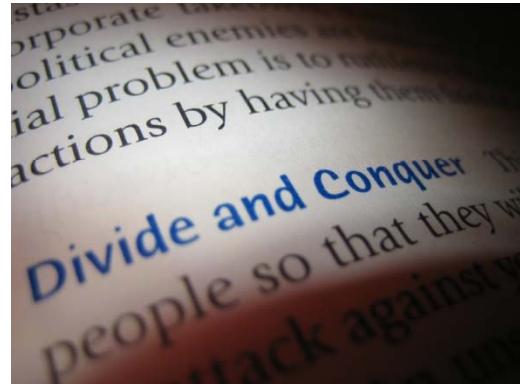
WATERFALL APPROACH

- The waterfall model originally came about
 - in complex but fairly stable development projects,
 - such as rocket systems, where NASA
 - needed to control requirements and schedules in great detail.
 - To NASA, not making changes that might create bugs
 - is far more important than being innovative or fast to market.



DIVIDE AND CONQUER

- In software, this means:
 - you should break large projects into multiple subprojects or milestones
 - of no more than a few weeks' or months' duration.
- It is much easier to manage several small groups
 - that are doing a smaller amount of work and
 - have a deadline that is only a few weeks or months away
- Than to manage one large group creating
 - a large number of features that
 - it is supposed to deliver in a year or more.



BEST PRACTICE

- *self-scheduling by the developers*
 - *not only produces an aggressive schedule,*
 - *which managers like,*
 - *it also produces both the appearance and the reality of being a fair schedule*
 - *because it comes from the bottom up.*
- In Microsoft: responsibility is not writing code, testing, specification writing, but shipping product.
- Managers need to keep historical data on projects and individuals
 - to judge the accuracy of the estimates and keep everyone honest.
 - commitment of people are very important, than dictating by managers

BEST PRACTICE

- What happened at both Microsoft and Netscape is that the rule requiring developers
 - to fix their own code if they broke the build
 - created a dynamic where people, on their own,
 - decided to check in very frequently-a couple of times a week and once a day or more toward the end of a project.
- In automobiles and other industries, we learned from Japanese companies decades ago that
 - it is ultimately cheaper to "build in" quality continuously
 - rather than to test and fix it in at the end of a de-velopment or production cycle.

BEST PRACTICE

- Over multiple projects, it is desirable to have a strategy to
 - improve process and product quality on a continuous basis-the
 - now-familiar Japanese notion of *kaizen*
- A phrase common at Microsoft and other companies—"eat your own dog food"

OUTSOURCING-1

- Many local entrepreneurs and foreign firms took advantage of:
 - the excellent technical education in India and
 - the English-language skills of their software engineers.
- suggests that outsourcing may be the best solution
 - for dealing with common problems
 - and high costs in software development.



OUTSOURCING-2

- One approach is for a customer's engineers to
 - do all or some of the analysis and high-level design work
 - architectures and functional specifications
 - and then subcontract the programming tasks
 - detailed design, coding, and unit testing
 - to offshore subsidiaries
 - E.g. MIEL, in the case of Mo-torola
 - or other firms such as Infosys



OUTSOURCING-3



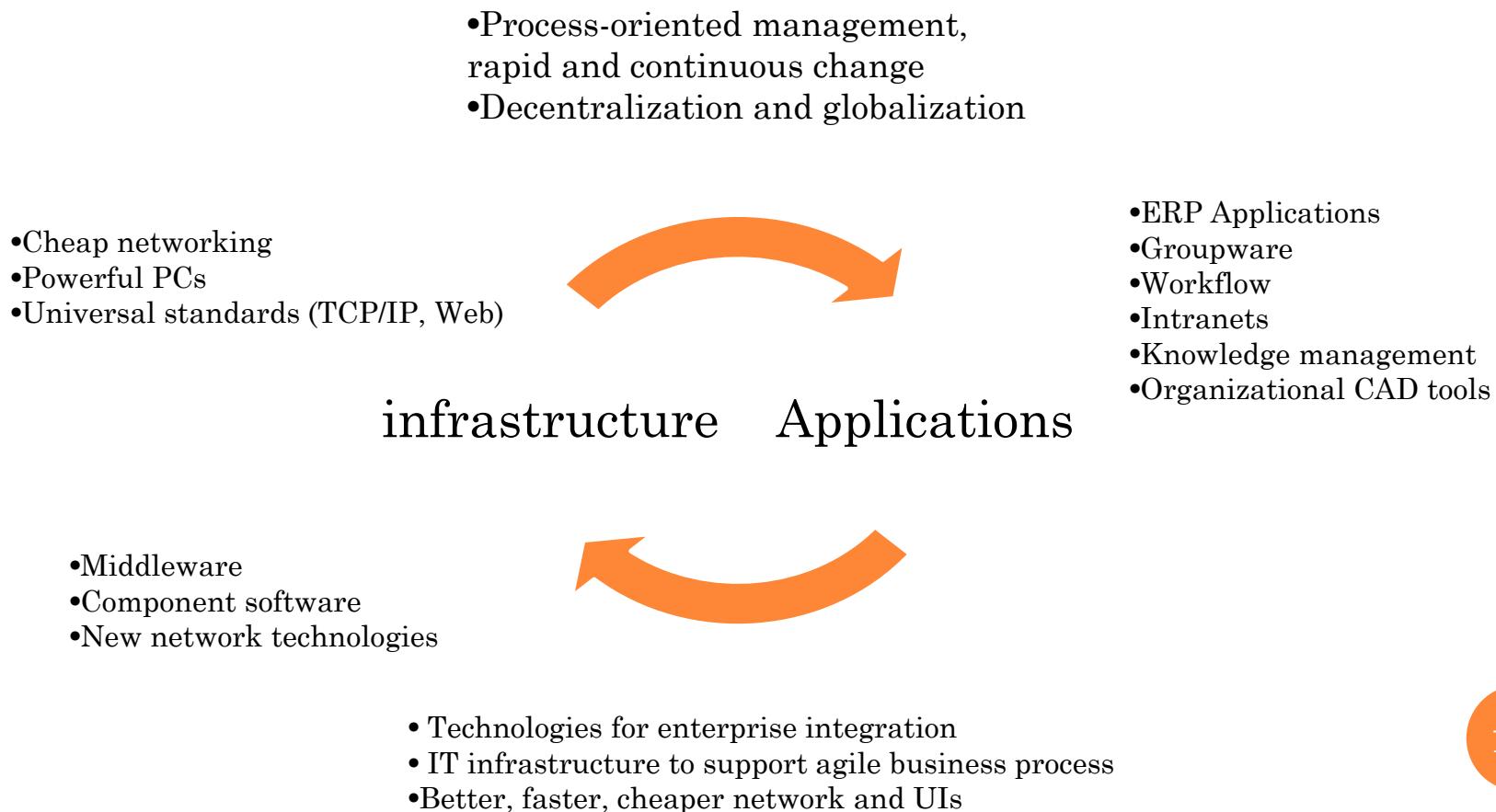
- The problem with the first type of software outsourcing is that:
 - it requires a "factory-like" separation of design from product construction
 - and usually results in a waterfall approach to project management.
- Outsourcing program construction is not the best way to do innovative software development
 - It can force projects into a rigid waterfall model
 - because it is easier to structure work sequentially
 - and takes more effort to do frequent builds
 - and incorporate customer feedback quickly if the development team is dispersed or separated from the customer.

FINANCING FOR NEW SOFTWARE BUSINESS

- In general, though, convincing anyone venture capital firm
 - to fund a particular idea
 - appears to be an extraordinarily difficult task
 - far more difficult than creating a successful company,
 - which is difficult enough.
- *You can't succeed without trying.*
 - Some start-ups will get the billions of dollars lying dormant in the coffers of venture capitalists,
 - and one of them may very well be yours.
- At the same time
 - accepting venture money when you don't need it,
 - or accepting more than you need,
 - can often become what is called "the kiss of death."



PREDICTING TRENDS IN INFORMATION TECHNOLOGY



OUTLINE-2

- Software Business Operations
- Software Business Entrepreneurship
- Open Source Software Initiation
- Open Source Business Models and Revenue
- Open Source Software Operations
- Commercial Open Source

WHAT ENTREPRENEURS SHOULD CONSIDER

- *Point 1: A Strong Management Team*
 - Many venture capitalists will tell you that they invest primarily in people
 - and secondarily in ideas
 - experience, commitment, marketing, development are very important
- *Point 2: An Attractive Market*
 - The potential market for a start-up's main product or service should
 - Be large enough, growing fast enough, or potentially profitable enough
 - to get the attention of outside investors.
- *Point 3: A Compelling New Product, Service, or Hybrid Solution:*
 - A start-up should offer something that is *compelling* to a specific type of customer.
 - That something can be a product, such as a new software development tool that many programmers are likely to find useful.

WHAT ENTREPRENEURS SHOULD CONSIDER

- *Point 4: Strong Evidence of Customer Interest:*
 - Entrepreneurs should present evidence to investors
 - that *actual customers are willing to buy the offering.*
- *Point 5: A Plan to Overcome the "Credibility Gap"*
 - One of the most difficult problems a start-up-even a well-funded start-up-has to overcome
- *Point 6: A Business Model Showing Early Growth and Profit Potential Wannabe*
 - entrepreneurs and their business plans should be able to describe
 - not simply an interesting idea, technology, or product concept,
 - but a proposal for *a growing and profitable business* at an early stage of the company,
 - such as within one or two years

WHAT ENTREPRENEURS SHOULD CONSIDER

- *Point 7: Flexibility in Strategy and Product Offerings:*

- Flexibility in both strategy and the product or service increases the chances that
- a start-up will find the right formula
- In the strategy field and in business history in general,
- we have seen many cases where success emerges only over time,
- through trial and error; success stories are rarely planned exactly in advance.



WHAT ENTREPRENEURS SHOULD CONSIDER

- *Point 8: The Potential for a large Payoff to Investors:*
 - The business plan should include prospects for a significant "payoff"
 - to outside investors and a reasonable return on investment
 - at least 25 percent
 - within a time frame that VCs are comfortable with based on the life span of their investment funds
 - typically, no more than seven years

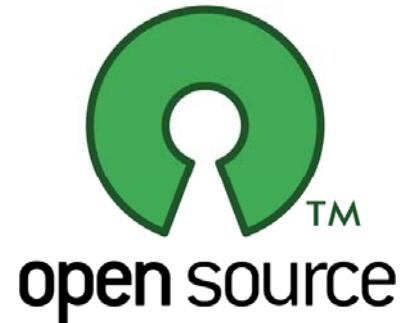


OUTLINE-2

- Software Business Operations
- Software Business Entrepreneurship
- Open Source Software Initiation
- Open Source Business Models and Revenue
- Open Source Software Operations
- Commercial Open Source

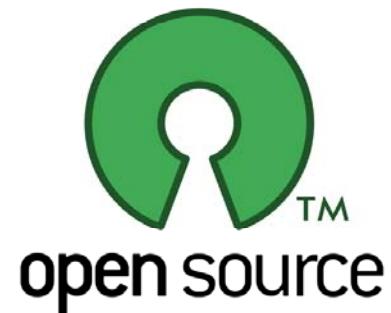
OVERVIEW OF OPEN SOURCE

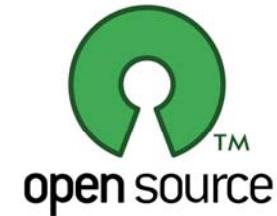
- Needed to make a computer perform a particular operation
 - If know the source needed
 - Simply enter such code in his/her operating system
 - The computer will respond as desired
- If individual can access the underlying source code
 - Then just copy the code
 - Not need to purchase the original program



OVERVIEW OF OPEN SOURCE

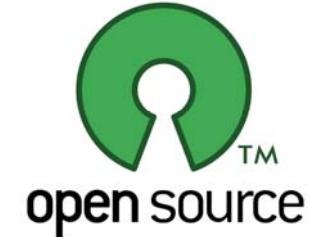
- For this reason many software companies:
 - Close off access to the source coding
 - User cannot readily access and copy the mechanism that makes the product work
 - Instead individual must use an interface
 - Allows them to active certain command indirectly
 - Called Proprietary software, for only creator/copyright holder, is allowed to open or see and copy or manipulate underlying source code.





OPEN SOURCE SOFTWARE

- *Open source software* (OSS), however, represents a completely opposite perspective in terms of:
 - access to source code
- OSS products are, in essence, created in such a way that:
 - access to the underlying source code is *open* and available for others to access and review
- A very basic yet common example of such open access to coding is:
 - HTML,
 - which allows browsers to display Web pages.
 - The coding of these pages is generally open for anyone to:
 - review and replicate
 - All one needs to do is access a page's underlying coding by using the
 - “View Source”—or other related option—in his or her browser.



OSS CHARACTERISTICS

- First, and perhaps foremost, because OSS is open for the user
 - to modify as desired,
 - it is easy for each individual to use the same programming foundation/source code
 - to develop different non-compatible softwares.
- Such divergence is often referred to as *forking code*
 - each programmer working on the development of an OSS item can take a different “fork” in the programming *road*, and
 - with each different fork, two programs that were once identical become increasingly different from one another

OSS CHALLENGES

- Such forking code, moreover, has long been considered a major problem in OSS development.
- These prospects for divergence mean OSS use is open to a variety of problems involving compatibility
- Such problems include:
 - Individuals generating software that others cannot use due to compatibility issues
 - Software that does not work as desired or work in unexpected ways
 - Parts of distributed programming projects not working as intended or not working at all
 - Users becoming frustrated with and abandoning software they consider too time-consuming or cumbersome to operate

OSS CHALLENGES

- Some companies, such as Linux, have addressed the problem of forking code and compatibility:
 - through focused oversight processes that govern programming practices.
 - The result has been successful and relatively stable software products that work effectively with other systems
- A second major problem area for OSS involves:
 - the technical support available to users of such software
- Because it is often the case that no individual or organization:
 - really *owns* an open source software product,
 - there is often no formal or standard mechanism for providing technical support to OSS users.

OSS CHALLENGES

- Such support instead tends to come from:
 - loose networks of OSS developers and
 - aficionados who interact informally in online contexts such as chat rooms or listserevs.
- Within this context:
 - *technical support* generally means a user who is:
 - experiencing difficulty posts a query to an online OSS forum and
 - then waits for a member of that forum to read the posting and reply.

OSS CHALLENGES

- One limitation of such an informal support system is that answers are not readily available
- Instead, individuals could find themselves waiting for anywhere from seconds to days:
 - from some random community member to respond
- Such delays could, in turn, have a major effect on:
 - the usability and the desirability of OSS product
 - not to mention the successes with which individuals can use such products to interact
 - Equally problematic is that such technical support systems are open for anyone to participate in and provide advice or solution
 - regardless of the technical skills of the individual.
- Thus, the quality of the advice provided by OSS support systems can be:
 - haphazard,
 - inconsistent,
 - or even incorrect.

FREE SOFTWARE

- The free software (FS) movement is:
 - the key predecessor of the open source (OS) community
- The FS movement, in turn, is based on:
 - arguments developed by Richard M. Stallman. (MIT AI lab),
 - The project was founded on a philosophy of software freedom, and the related views on copyright or the concept of *copyleft*.
- The FS movement has laid:
 - technological, legal and ideological cornerstones
 - that still exist as part of the open source movement.



FREE SOFTWARE



- In *Hackers* (1984), Steven Levy describes:
 - the subculture around the AI lab computers in the 1960s
 - Young male electronics hobbyists devoted their time to programming and studying these machines
- They called themselves *hackers*, a word denoting:
 - a person who enjoys exploring computer systems,
 - being in control of the systems, and
 - facing the challenges they present
- For a hacker, a computer:
 - is not just a tool, it is also an end in itself
- The computer is something to be respected and programming has an aesthetics of its own

HACKER ETHICS

- The six rules of this *hacker ethic* as later codified by Levy were:
 - *Access to computer and anything:*
 - *which might teach you something about the way the world works*
 - *should be unlimited and total. Always yield to the hands-on imperative!*
 - *All information should be free.*
 - *Mistrust authority*
 - *promote decentralization.*
 - *Hackers should be judged by their hacking,*
 - *not bogus criteria such as degrees, age, race, or position*
 - *You can create art and beauty on a computer.*
 - *Computers can change your life for the better.*



FREE SOFTWARE HISTORY

- Computer programs were treated like:
 - any information created by the scientific community
 - Software was free for everyone to use, study, and enhance
 - Building on programs created by other programmers was not only allowed, but encouraged
 - On one hand, nobody owned the programs,
 - and on the other, they were common property of the community.
- The community and its way of life had been destroyed and Stallman later described himself as “the last survivor of a dead culture”



STALLMAN SAID:



I consider that the golden rule requires that if I like a program I must share it with other people who like it. Software sellers want to divide the users and conquer them, making each user agree not to share with others. I refuse to break solidarity with other users in this way. I cannot in good conscience sign a nondisclosure agreement or a software license agreement...So that I can continue to use computers without dishonor, I have decided to put together a sufficient body of free software so that I will be able to get along without any software that is not free.

...copyright licensing is an easy way to make money but is “harming society as a whole both materially and spiritually.”

FREEDOM SOFTWARE DEFINITION

- In *Free Software Definition* (Stallman, 2002), listed the *four freedoms* which a piece of software must meet:
 - **Freedom 0:** The freedom to run the program, for any purpose
 - **Freedom 1:** The freedom to study how the program works, and adapt it to your needs; access to the source code is a precondition for this
 - **Freedom 2:** The freedom to redistribute copies so you can help your neighbor
 - **Freedom 3:** The freedom to improve the program, and release your improvements to the public, so that the whole community benefits; access to the source code is a precondition for this
- Stallman makes an important distinction between:
 - *free as in free speech* and *free as in zero price*
 - The concept of free software is not against selling software
 - it is against restrictions put on the users
 - Free software can be sold but the seller may not forbid the users to share or modify it

OPEN SOURCE VS. FREE SOFTWARE

- The open source movement is largely composed:
 - not of people who reject Stallman's ideals,
 - But rather of people who reject his rhetoric
- Thus, the aim of the term *open source* is:
 - to emphasize the practical benefits of the OS development model
 - instead of the moral philosophy behind the free software ideal
- For the actors in the OS movement, the creation of OS software is:
 - An utilitarian venture of collaboration
 - based on individual needs

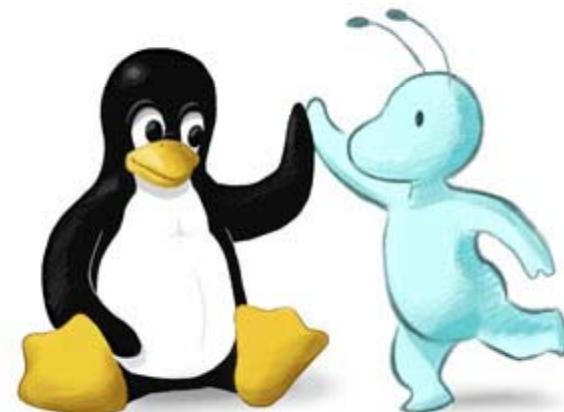
FIRMS PARTICIPATION IN OPEN SOURCE COMMUNITIES

- Well-known examples of firms that use the communal resources of OSS communities are:
 - IBM, SUN, and Red Hat
- These and other organizations hope to benefit from OSS
 - because they believe it constitutes a low-cost and high quality knowledge resource
 - that may spur new product development
- They believe characteristics of OSS communities,
 - like the release of source code,
 - may provide opportunities that lead to:
 - the early adoption of new products and
 - hence lead to first-mover advantages



ENGAGEMENT OF COMMERCIAL ORGANIZATIONS

- Engagement of commercial organizations in OSS communities may also provide various benefits to OSS communities, since firms may:
 - enlarge the user base of the communities,
 - contribute scarce financial and human resources, and
 - perform a boundary spanning function by linking the communities to various groups of non-technical users.



CONCEPT OF GIFT ECONOMY

- Why do people participate even when
 - the efforts, or costs, involved in writing source code
 - or solving other people's problems
 - do not exceed the direct monetary benefits
 - that can be gained from such activities
- Dominant answer: gift economy
 - Which individual like to give.
- The concept of a gift economy can be traced back to Mauss (1990):
 - the giving of gifts laid the foundation for exchange
 - A *gift economy* relies on the principle of reciprocity and an implicit requirement to give
- In these systems “a gift is not so much a physical resource
 - as a social and moral system by which sharing,
 - collaboration, loyalty and trust are cultivated
-



CONCEPT OF GIFT ECONOMY



- Indeed, there are some indications that the principle of gift giving is important in OSS communities
 - “Open-source contributors have told us that they enjoy the sense of ‘helping others out’ and ‘giving something back’”
- Answer of participants:
 - “It is nonsense to believe that in OSS you do not receive anything If you do what you are good at, others will do the same. I receive a lot from others, which I could not have done myself. In the gift economy everybody is better off.”
 - participants in the communities are said to create and sustain dynamic relationships with one another based on the exchange of gifts

MOST FREQUENTLY IDENTIFIED BENEFITS

- Some of the most frequently identified benefits of participating in OSS communities are:
 - Building a reputation in a community
 - Learning and improving one's programming skills
 - Meeting a personal need with a software program that has a certain functionality
 - Having fun

OUTLINE-2

- Software Business Operations
- Software Business Entrepreneurship
- Open Source Software Initiation
- Open Source Business Models and Revenue
- Open Source Software Operations
- Commercial Open Source

FIRMS PARTICIPATION IN OPEN SOURCE

- reasons for firms to participate are:
 - OSS communities allow small firms to be innovative,
 - contributions and feedback from the OSS communities are very useful to fix bugs and improve software, and
 - open source software is reliable and has a high quality
- OSS is currently used by many private and public organizations. Good examples are:
 - New York Stock Exchange,
 - Shell,
 - the French daily Le Figaro,
 - the U.S. army,
 - national government in Brazil, and
 - the city of Munich



HOW FIRMS PARTICIPATE IN OPEN SOURCE

- Most firms generate the majority of their revenues from open source solutions,
 - but generally combine these with more traditional proprietary offerings
- Interviews with the business founders suggested that:
 - many customers are still unaware of open source or
 - perceive it as a risky alternative and as a consequence,
 - many firms are more or less “forced” to also offer proprietary solutions



HOW FIRMS PARTICIPATE IN OPEN SOURCE

- With regard to product offerings, we founding suggests:
 - most firms sell little hardware solutions
 - Sales of a typical firm are based for about 41% on software development,
 - while over 52% of revenues come from offering additional business services such as consultancy, support, and training
- Interestingly, this pattern mirrors the business models of the more traditional Dutch IT firms
 - that also predominantly generate revenues from selling IT services.

HOW FIRMS PARTICIPATE IN OPEN SOURCE

- This finding can be explained by the observation that:
 - most firms rely for a large part on selling OSS-related services,
 - which are generally more dependent on geography
 - and more locally oriented than firms selling software packages that can be distributed across foreign markets
- An alternative explanation could be that:
 - technical activities like contributing source code and answering questions on mailing lists create international recognition for the firm
 - This recognition in turn may create international demand and thus foreign sales.

HOW FIRMS PARTICIPATE IN OPEN SOURCE

- firms in which the founding members have experience in the IT industry:
 - perform fewer technical activities in OSS communities
 - as compared to firms in which the founding members do have experience in marketing and sales
- More IT experience implies:
 - less technical participation and more social participation.

HOW FIRMS PARTICIPATE IN OPEN SOURCE

- The activities that firms undertake in OSS communities can be grouped into two distinct categories:
 - A first group of activities, referred to as social participation, includes activities like organizing workshops and conferences
 - The second set of activities, which we label technical participation, includes actions such as contributing source code, bug fixes, and participating in mailing list discussions
- One important result concerns the finding that firms seem to view their internal investments in R&D as:
 - a complement to their external product-development activities in OSS communities.

OPEN SOURCE VS. PROPRIETARY SOFTWARE

- Another clear distinction between open source and proprietary processes can be found in the type of the process itself:
 - There is the *disclosure feedback approach* used in open source projects,
 - and the *secrecy-incorporation approach* used by traditional firms in the software industry
- Not only do these two approaches require different communication paths between end-users and developers
 - they also influence the complete organizational culture of a software community.

OPEN SOURCE VS. PROPRIETARY SOFTWARE

- It can be argued that institutional secrecy incorporation culture might cause:
 - a built-in tendency for the vendor to focus on the code and
 - the feature list from his own point of view only,
 - growing a product that from the outside might be what customers think they want,
 - but that from the inside slowly turns into a dinosaur



BEING COMMUNITY CUSTOMER

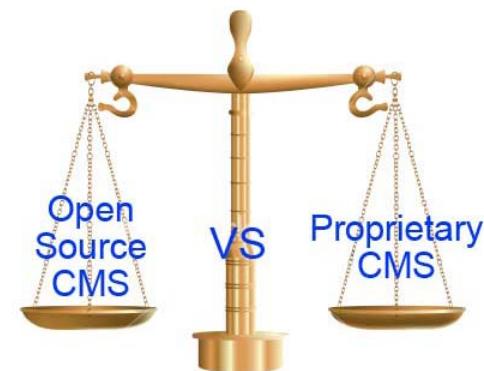
- Two main aspects of open source community participation can be distinguished:
 - *rent-seeking* and
 - *donation*.
- Donation: when the end-using organization sees the open source community itself as the vendor, and becomes *a customer of the community*:
 - This is distinctively different from free riding on the product
 - the organization truly spends resources on the customership
 - In many cases, there is no monetary exchange between the organization and the community,
 - as often the community has no central representative which accepts money in exchange for services, such as with a traditional vendor (even if hybrid)
 - But the customer certainly may spend resources on the open source product, by *donating effort around the product to the community*

BEING COMMUNITY CUSTOMER

- A very common way of becoming a community customer is:
 - to pay employees or contractors to do some work around the product,
 - which is not necessarily development,
 - and then instructing them to donate the results to the community
- The key difference may be that:
 - open source culture favors the *process* over the *product*
 - Typical proprietary culture favors the *product* over the *process*
- community customers must focus on:
 - the process, not on the product, and that their donations should be aimed at the process.

BEING COMMUNITY CUSTOMER

- The term *customer* should be assumed to mean exactly that: offering resources, monetary or otherwise, to receive services or products in exchange
- Summarizing, active participation in the development process is true community customership for organisations,
 - but by no means the only possibility



HOW DO OPEN SOURCE COMPANIES MAKE MONEY?

- It is true that an open source business may not make money directly from its products,
- It is untrue that open source companies do not generate stable and scalable revenue streams
- In actuality, in the 21st century web technology market:
 - it is the open source company that has the greatest long-term strategic advantage

HOW DO OPEN SOURCE COMPANIES MAKE MONEY?

- This is demonstrated by companies such as:
 - LINUX,
 - Apache, and
 - Netscape,
 - a host of web-specific technologies such as Java, Perl, TCL, and a
 - host of web-specific technology companies such as Sendmail



OPEN SOURCE BUSINESS MODEL

- Relies on:
 - shifting the commercial value away from the actual products and generating revenue
 - from the 'Product Halo,' or ancillary services like systems integration, support, tutorials and documentation.
- This focus on the product halo is rooted in:
 - the firm understanding that in the real-world, the value of software lies in the value-added services of the product halo and
 - not in the product or any intellectual property that the product represents.
- In actuality, the value of software products approaches zero
 - in the fast-paced, highly-customized, ever-changing world of information technology

OPEN SOURCE BUSINESS MODEL

- But it is not simply an acknowledgement of the revenue streams:
 - generated by the product halo that makes open source a compelling business strategy
- Open source also:
 - cuts down on essential research and development costs
 - while at the same time speeding up delivery of new products



OPEN SOURCE BUSINESS MODEL



- This paradoxical situation arises from the fact that:
 - within an open source project, the community members themselves provide free research and development
 - by contributing new solutions, features, and ideas back to the community as a whole
 - The company that sits at the center of any successful open source project
 - may reap the rewards of the work of thousands of highly-skilled developers without paying them a cent.

OPEN SOURCE BUSINESS MODEL

- A final strength of the open source business model lies in:
 - its ability to market itself
- Because open source products are typically released for free:
 - open source companies that can produce quality products and
 - generate a good reputation
 - can almost immediately grab huge shares of any market
 - based on the complex and far-reaching global referral networks generated by users

OPEN SOURCE BUSINESS MODEL

- In the web technology space, almost every global standard has been based upon open source technology
- By using the open source technology model:
 - we can create a superior product,
 - which immediately has a competitive advantage, and
 - which generates multiple scalable revenue streams while being freely available throughout the community

OPEN SOURCE BUSINESS MODEL

BUSINESS WEEK OVERVIEW

- While the open-source business model may be broken:
 - the concepts behind open source will continue to bring
 - new value to customers and strong returns to software company stakeholders
- But the value is in:
 - the collaboration, not in open source itself

OPEN SOURCE BUSINESS MODEL

BUSINESS WEEK OVERVIEW

BusinessWeek

- Think about it like going in with others on a pizza:
 - Too often, businesses need to develop software with the same "ingredients" as everyone else, and
 - this offers no competitive advantage
 - If everyone wants the same pizza, why not share the cost? And by the way,
 - let's not just share the cost; let's make it together so we get it just right and know what we're getting

OPEN SOURCE SOFTWARE



- Revenue Models in the Open Source Software Business Summarizing the ideas behind the terms in:
 - Open Source Distribution (OSD)
- OSD says the software license must generate the following effects:
 - Source code must be readable and available, either included with the binary code or publicly downloadable
 - Free distribution of the software by any party, on any medium, to any party, gratis or for a fee
 - Derivative works must be allowed, either under similar license or not, depending on the specific OSS license type
 - No discrimination against persons, groups, or fields of endeavor

CHARACTERISTIC OF VARIOUS BUSINESS MODEL ELEMENTS

- These elements, expressed in different words by different authors, include the following:
 - Offerings
 - resources needed to develop and implement a business model; and
 - relationships with other actors
- These elements are interconnected with
 - The revenue model,
 - including sources of:
 - revenue,
 - price-quotation principles, and
 - cost structures,
 - which is characteristic of a particular business

SUMMERY OF OSS REVENUE MODELS-1

Revenue model	Description	License Type	Revenue sources
Support selling	A for-profit company provides support for a software that is distributed free of charge	Any	Revenue comes from media distribution, branding, training, consulting, customer development, and post-sales support for physical goods and services
Loss-leader	A no-charge open source product is used as a loss leader for traditional commercial software (i.e., the software is made free by hoping that it will simulate demand for a related offering of the company).	Varies	Complementary offering (e.g., other software products)
Widget-frosting	Companies that are in business primarily to sell hardware can use this model to enable software such as driver and interface code. By making the needed drivers open, the vendor can ensure that they are debugged and kept up to date	Any	The company's main business is hardware. This is quite similar to the loss leader model.
Accessorizing	Companies that distribute books, computer hardware and other physical items associated with and supportive of OSS	Any	Supplementary offerning.

SUMMERY OF OSS REVENUE MODELS-2

Revenue model	Description	License Type	Revenue sources
Service enabler	OSS is created and distributed primarily to support access to generating revenue from consulting services and online services.	Any	Service fees
Brand licensing	A company charges other companies for the right to use its brand name and trademarks in creating derivative products	Storage reciprocity	Copyright compensations
Sell it, Free it	A company's software products start out their product life cycle as traditional commercial products and then are converted to open source products when appropriate	Alteration of license type	Initial revenue from software product offerings converted into other models (e.g. the loss-leader model)
Software franchising	A combination of several of the preceding models (in particular, brand licensing and support sellers) in which a company authorizes others to use its brand names and trademarks in creating associated organizations doing custom software development, in particular, geographic areas or vertical markets.	Strong reciprocity	The franchisor supplies franchisees with training and related services in exchange for franchising fees of some sort.



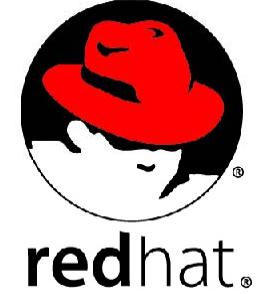
MySQL REVENUE MODEL

- The Revenue Model MySQL AB is often cited as:
 - the champion of the second generation of open source projects
- These projects are open source but:
 - are directed by for-profit companies
- The revenues of these corporations derive from:
 - selling consulting services for their products
 - MySQL AB makes MySQL available:
 - under the GPL for free and
 - sells it under proprietary licenses for clients
 - when the GPL is not an ideal option (
 - e.g., inclusion of MySQL technology in a closed source product

MySQL REVENUE MODEL



- Currently, MySQL AB receives more income from:
 - proprietary license sales
 - than from its other income sources, branding, and services
- Its main income seems to come from embedded commercial users
- In terms of Hecker (1999), the revenue models of MySQL AB include:
 - features from support selling and dual licensing,
 - both of which can be considered incarnations of the loss-leader model.



RED HAT REVENUE MODEL

- The Revenue Model Despite the release of software under the GPL license mode,
- The services employed by Red Hat for commercial viability places
 - a layer of restriction upon the binary and source code usage based on support contracts
- This hybrid approach enables the company
 - to provide OSS solutions in a commercial way (Microsoft, 2005)
 - Thus, the primary revenue model is currently what Red Hat calls “subscriptions,”
 - which allows the company to effectively develop and deliver its technology based on customer feedback,
 - as well as to provide support to customers over the life of an agreement.



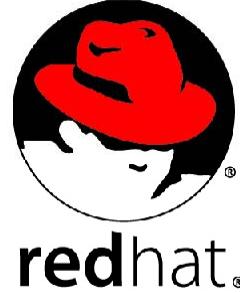
RED HAT REVENUE MODEL

- In terms of Hecker (1999), we identify this revenue model as support selling.
- It has been claimed that this is:
 - high-margin activity demanding only a little investment
 - little investment means lower entry barriers, and
 - support offers a very weak basis for differentiation to gain sustainable competitive advantage
- Microsoft clearly has nearly a monopoly on desktop operating systems,
 - but its market share in services related to desktop operating systems is much smaller
 - Thus, there is potential for revenue models based on service provisioning, as in some OSS-based businesses

SUMMERY OF MYSQL AND REDHAT BUSINESS MODEL

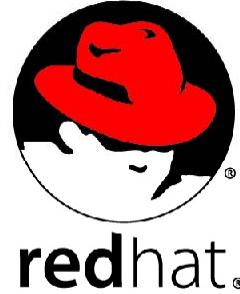
Business Model Elements	MySQL	RedHat
Offering	Core offering embodies an in-house development database software product and related services	Operating system software maintenance and services along with operating system software
Resources	The internal programming resources and professional management resources	Resources related to the development and management of brands, as well as to marketing and business management
Relationships	Balance between the OSS community and commercial business networks that have somewhat disparate needs and values. Dependence on the OSS community mainly as a user community.	OSS community as significant product developer
Revenue model	A majority of revenue originates from proprietary license sales, and a smaller proportion stems from other sources such as services. The main income seems to come from business users. The revenue model includes features from support selling and dual licensing, both of which can be considered incarnations of the loss-leader model.	The primary revenue model is currently what Red Hat calls “subscription”, which allows the company to effectively develop and deliver its technology based on customer feedback, as well as to provide support to customers over the life of an agreement. The revenue model is identified as support selling.

MySQL vs. REDHAT



- empirical observations from the two case examples indicate:
 - The selection of the revenue model is dependent on other business model elements
 - The case of MySQL illustrates that the need to maintain relationships with both:
 - the OSS community and
 - the business network
 - has led to a revenue model based on dual licensing
 - In this model, the community has access to the software for free,
 - but business users may buy a software license for their commercial purposes

MySQL vs. Redhat



- empirical observations from the two case examples indicate:
 - The dual-licensing model used by MySQL illustrates that:
 - change in any of the elements of the identified key determinants
 - may affect the revenue model choice
 - In this model, the company owns all copyrights to the software and,
 - therefore, can license the software with two licenses,
 - one allowing gathering of revenue from sold copies of the software
 - and the other based on the principles of the loss-leader model

BUSINESS MODELS IN OPEN SOURCE SOFTWARE VALUE CREATION

- Although the concepts of a business model and strategy are highly complementary, they are not the same
- A strategy focuses on value appropriation,
- while a business model explains how value is created for all stakeholders
- Chesbrough and Rosenbloom (2002) made three clear distinctions between the two.
 - First, a business model is based on value creation for the customer, but emphasis on capturing that value and sustaining it is part of the scope of a strategy.
 - Second, financing of the value creation is implicitly assumed in business models, whereas a strategy explicitly considers the financing issues of value creation
 - because of the underlying assumptions of shareholder value creation
 - Finally, there is a difference in the assumptions about the state of knowledge held by the firm and its stakeholders.

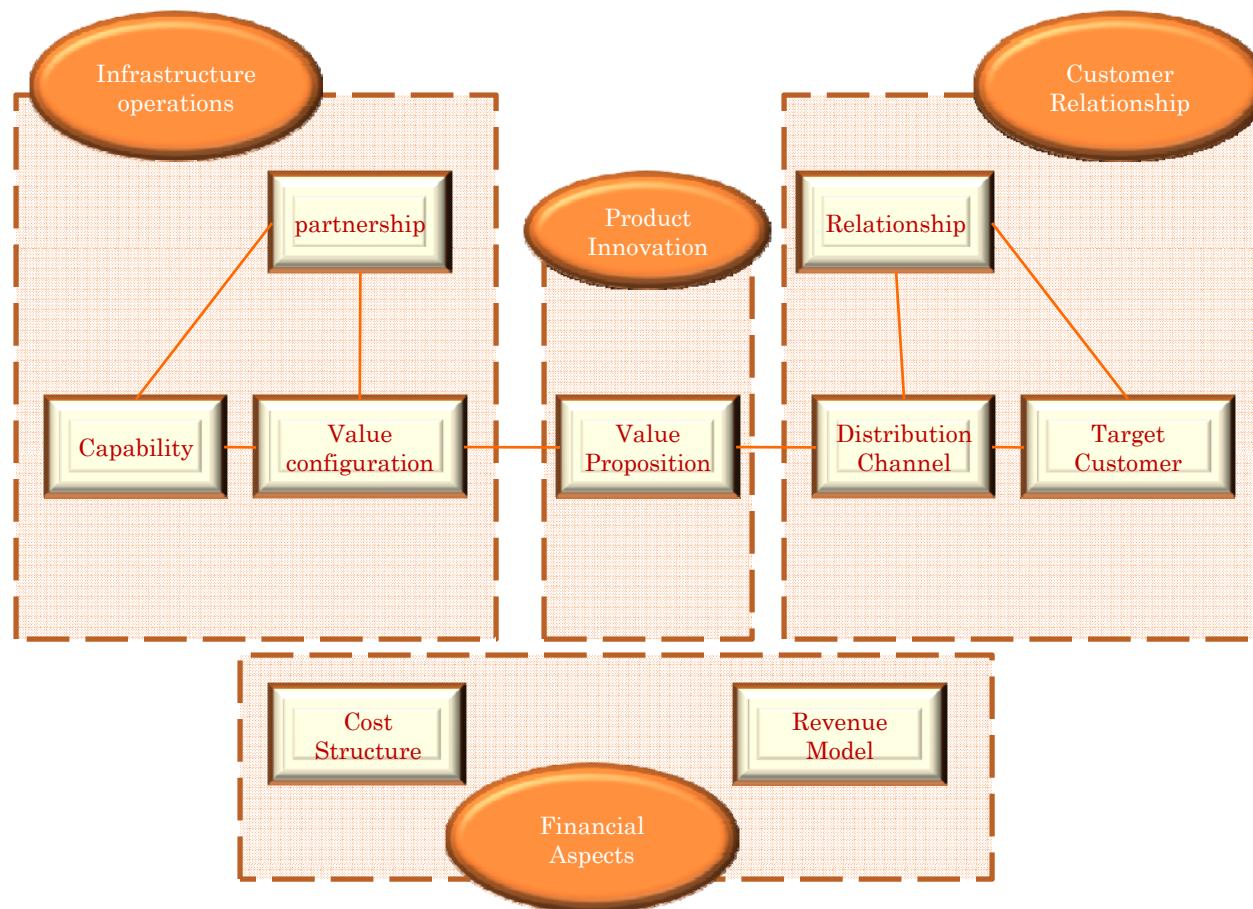
BUSINESS MODELS IN OPEN SOURCE SOFTWARE VALUE CREATION

- Business models consciously assume limited and distorted information and knowledge,
 - while a strategy is built on analysis and refinements in knowledge and,
 - therefore, assumes the existence of a plentitude of reliable information to be transformed into knowledge
- A practical distinction describes business models:
 - as a system that shows how the pieces of a business fit together,
 - while strategy also includes competition

PURPOSE OF BUSINESS MODEL

- We specify the purposes of a business model in accordance with the view of Chesbrough and Rosenbloom (2002), functions of a business are:
 - To articulate the value proposition
 - To identify a market segment
 - To define the structure of the value chain within the firm
 - To estimate the cost structure and profit potential
 - To describe the position of the firm within the value network
 - To formulate the competitive strategy

ELEMENTS AND STRUCTURE OF A BUSINESS MODEL



BUSINESS MODEL ELEMENTS AND EXAMPLE QUESTIONS IN OSS-1

Element	Description	Question
Value proposition	Given an overall view of the company's bundle of products and services	Does the utilization of OSS affect the way the customer perceives our value offering? How do we take OSS into account in customer marketing?
Target Customer	Describe the market segments to which a company wants to offer value	Who are our target customers? Does OSS impose restrictions or provide wider access for certain market segments?
Distribution Channel	Describes the company's the various means of getting in touch with its customers	Could we use open distributions (Sourceforge or other), our own web site, or other servers?) How are potential utilizers going to find us or our product?
Relationship	Explain the kind of links a company establishes between itself and its different customer groups	What is an appropriate OS license to use? What kind of relationship are we going to create with the community?
Value Configuration	Describes the arrangement of activities and resources	What is our role in the community? How do we share resources and carry out activities with the other actors and community players?

BUSINESS MODEL ELEMENTS AND EXAMPLE QUESTIONS IN OSS-2

Element	Description	Question
Core Competency	Outlines the competencies necessary to actualize the company's business model	What are the competencies we especially seek from and can offer to the community? How are we to manage relationships and maintain sustainable development?
Partner network	Portrays the network of cooperative agreement with other companies that are necessary for efficiently offering and commercializing value	What kinds of agreements are we going to make with various participants? How does utilization of OSS affect our partners outside the community?
Cost Structure	Sums up the monetary consequences of the means employed in the business model	What kind of cost structure do the aforementioned choice involve? Are we able to cope with the economic consequences?
Revenue Model	Describes the way a company makes money through a variety of revenue flows	What revenue model should we choose? Do we prepare revenue and risk sharing models.

NOVELL



- *Novell, Inc:*
 - *was a leading network operating system provider in the 1980s and early 1990s*
 - *in the mid-1990s, Novell lost market share in the network operating system market*
 - *To counter this loss of market share, Novell made a strategic decision to go open*
 - *i.e., to make use of open standards and open source business strategies*
- Novell is believed to have effectively entered the service market and is considered:
 - a successful open source provider,
 - having followed a systematic rather than a “big-bang” approach

CATEGORIZATION OF OPEN SOURCE BUSINESS MODELS

- Hawkins (2004) asserts that open source business models can be subdivided into two categories:
 - business models for the software consumer and
 - business models for the software producer.
- When referring to the models for consumers, this signifies:
 - the total cost of ownership (TCO) of the chosen software solution

CATEGORIZATION OF OPEN SOURCE BUSINESS MODELS

- When referring to the models for software producers and, in particular, the revenues of the company:
 - there are a few prospective sources of revenues, such as:
 - sale of software,
 - support of software,
 - increased hardware sales,
 - training, consulting,
 - customization, distribution, and
 - the value of internal use

NOVEL

Novell.

- The subscription strategy, also known as the revenues-for-services strategy:
 - is one in which a provider charges a license fee for software
 - mainly to provide maintenance and consultation services.
- Novell uses this strategy.
- Novell acquired SuSE (Software- und System-Entwicklung):
 - in an attempt to supplement its declining NetWare maintenance revenue
 - and to enter the Linux desktop market in which the adoption rate is very promising

NOVEL



- Approach that provides a way to sell and make money with OSS:
 - the trick is not to sell a support contract but rather an administration contract
 - users of systems do not need support all that often but do require their systems to be administered on a regular basis
 - Users of computing systems require their computers to be updated with the latest security patches and application updates,
 - something users do not want to do or do not have the relevant experience or knowledge to do

OPEN SOURCE SERVICE PROVIDERS

- On the other hand, Hohensohn and Hang (2003) maintain that open source service providers can be subdivided into five categories:
 - distributors as OSS service providers,
 - large hardware producers,
 - large software firms,
 - global system integrators, and
 - specialized open source service providers
- According to an internal McKinsey consultancy study:
 - 30% of the income from enterprise solutions comes from license fees and
 - about 70% from implementation of the solution

OPEN SOURCE SERVICE PROVIDERS

- In addition, a 2000 U.S. Department of Commerce report states that:
 - not since 1962 has software package cost exceeded 30% of the total software investment (
- In line with this, Novell's software license net revenue for 2004 and 2005 was:
 - 25% and 22%, respectively
- This confirms that the other 70+% of the software investment goes:
 - toward consultation,
 - maintenance, and
 - other related services.

OUTLINE-2

- Software Business Operations
- Software Business Entrepreneurship
- Open Source Software Initiation
- Open Source Business Models and Revenue
- Open Source Software Operations
- Commercial Open Source

OPEN SOURCE PROCUREMENT



- Although there are no licensing costs associated with open source products, their procurement costs are not zero
- There is evidence that open source product assessment and selection might be significantly more expensive (for the customer)
 - than proprietary assessment and selection,
 - as much of the actual cost is shifted from the vendor to the customer
- Using industry-average data, it was estimated that the sale of a proprietary learning system would cost the vendor:
 - over US\$250,000 in proposal writing, large-scale demonstrations using detailed, prescribed scripts from the customer, expert presentations

OPEN SOURCE PROCUREMENT



- All this work needs to be done by other people (not from a vendor) :
 - when the procurement of an open source system is investigated.
 - A part can come from documented community experiences,
 - but it mostly is up to the customer to spend the resources
- Although, obviously, the customer ends up paying the costs back to the vendor in case he decides to license the proprietary product,
 - he does not need to pay in case he does not license the product
- This no-purchase-no-pay option is unavailable when investigating open source software,
 - as there is nobody else who takes the risk of spending the money in the hope of winning a sale and getting it back with profit

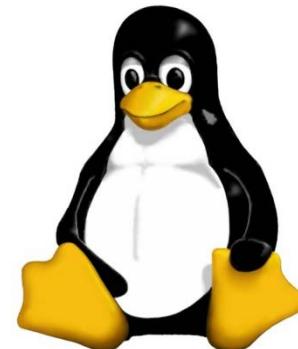
OPEN SOURCE PROCUREMENT



- open source procurement must:
 - be approached differently than proprietary procurement
- Initially an organization needs to invest more of its time and resources,
- but the open source community will be inclined to help, and even more so
 - if the organization shows respect by donating its experiences back to the community straight away
 - preferably not after the whole procurement process has been completed, but much earlier

OPEN SOURCE PROCUREMENT

- If the organization decides to deploy the open source product,
 - it will be immediately rewarded for its donation
 - by not having to pay the marketing and sales efforts back to the vendor in licensing costs
 - Instead, it has invested soundly in its community reputation
 - It has become a true community customer.



DEPLOYMENT PROCESS

- The community roles in the deployment process :
 - are for a large part the same as those in the procurement process.
- Documented:
 - previous experiences,
 - best practices,
 - how-to and
 - other helpful guidelines
 - are a valuable resource for any deployment project.



DEPLOYMENT PROCESS

- These resources need to be built up by organizations that deployed the product in the past
- Mature open source products usually have a significant body of this type of documents available
- Lakhani and von Hippel (2003) summarize all these resources plus personal help:
 - as *field support*, and call it essential for open source project success.



DEPLOYMENT PROCESS



- time spent on reading these resources may average 100 hours per year for active participants.
- Many of them then get into the habit of actually replying to cries for help from others “while they are there anyway”
- This is not time lost to charity; it is a sound investment in their organization's visibility in the community
- It will be noticed by other community members and when the organization needs help itself, it will get it
- As a community customer, they paid for it

DEPLOYMENT PROCESS

- Operational issues around software are not much different for both proprietary and open source products
- Both need regular patching for bugs and for security problems
- Both need to have feature development going on,
 - as no environment stays the same for very long
- The community provides these patching services as:
 - a natural part of the process, and
 - the organization using the open source product must be as committed to keep their installation up to date as with a proprietary product



DEPLOYMENT PROCESS

- What may be different is that open source products tend to have a livelier patch cycle
- Proprietary products typically have many months:
 - between releases and may provide an update service via the sales organization,
 - actively approaching their known customers with patches depending on the perceived urgency
- Open source products may offer the same service,
 - but usually do not actively approach customers
 - Instead, the community relies on being actively monitored

DEPLOYMENT PROCESS

- With the current trend towards online updates,
 - many products from both proprietary and open source origin check for updates automatically,
 - and even may apply the patches automatically without service interruption.



FIRMS' PARTICIPATION IN OPEN SOURCE

- This open community steering means two things for organizations that deploy open source products:
 - they have a heavier vote
 - in the product's development if they are actively participating in the community, and
 - if they are serious about some required feature which is not getting enough attention, they may develop it
 - or have it developed by a third party and
 - donate it to the community for further integration and maintenance



FIRMS' PARTICIPATION IN OPEN SOURCE

- In the end, because of resource sharing and economies of scale:
 - the result often will be obtained with less overall resource spending than with classical proprietary production
 - where competition is the main driving force behind development
 - A side effect is a reduced chance that a critical product suddenly disappears from the market due to competition
 - It is much more likely that a timely course change takes place, or a friendly merger with another product that appears better designed or uses newer technology



CUSTOMER AND USER INVOLVEMENT

- *From a strategic perspective, open source falls into:*
 - *category of business models that generate advantages based on customer and user involvement (CUI)*
- *Open source has been a novel strategy in the software business:*
 - *CUI-based strategies have been used elsewhere before*
- *Since the success of e-commerce and e-business:*
 - *CUI-based strategies have become far more prevalent for at least two reasons:*
 - *Firstly, advances in information technology and systems have improved feasibility of implementation of CUI strategies and*
 - *secondly, CUI-based economics appear to have often become a requirement for e-business profitability*

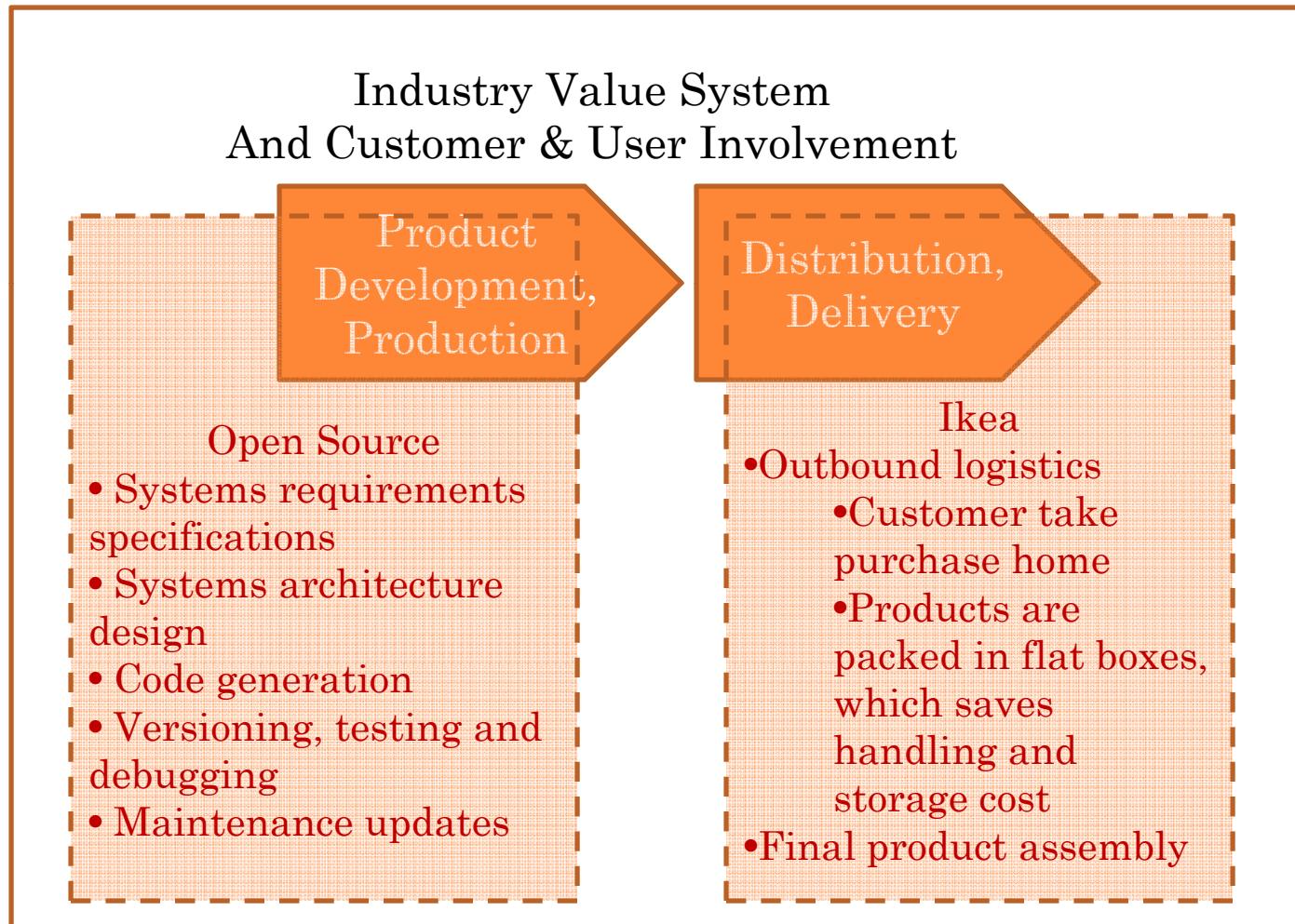
CUSTOMER INVOLVEMENT



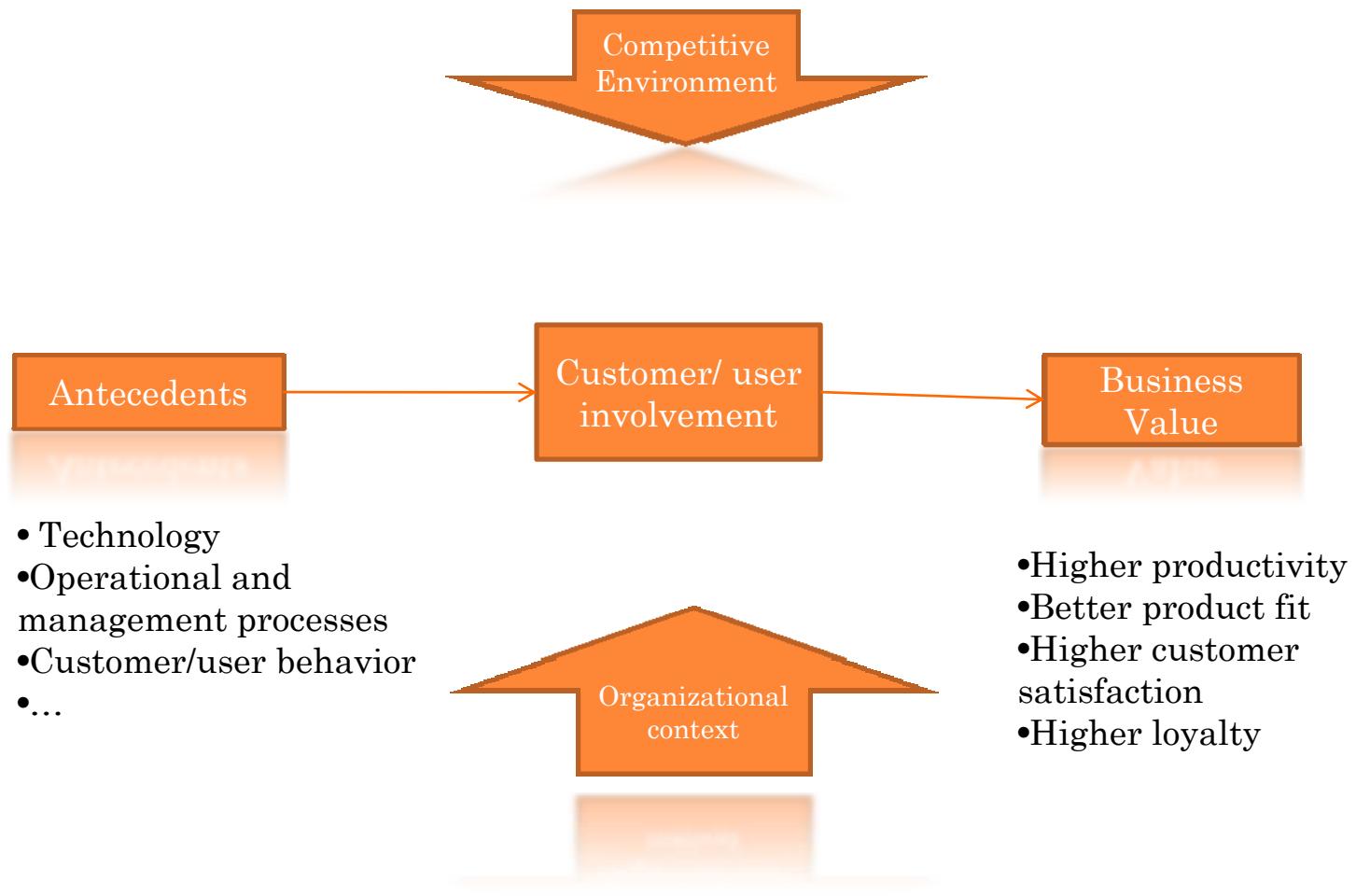
- Customer involvement economics can be an enabler of other economic advantages.
- In the Ikea example:
 - The cost advantage due to customer involvement is used or leveraged by
 - splitting savings with the customer,
 - effectively lowering product prices,
 - often below the price of the competition
 - The lower sticker price makes stylish design affordable for a larger market,
 - which increasing Ikeas market potential
 - This larger footprint, in turn, allows Ikea to benefit from
 - another economic advantage, the one that has been the main economic engine of mass production, namely scale economies.

CUSTOMER AND USER INVOLVEMENT

COMPARISON OF IKEA AND OPEN SOURCE



THEORETIC CUSTOMER AND USER INVOLVEMENT MODEL



CUSTOMER INVOLVEMENT

- The literature defines customer involvement as:
 - the extent to which a customer is engaged as a participant in business operations,
 - specifically in service production and delivery, including,
 - E.g. order processing and account management.
- The customer involvement construct and its definition are rooted in several streams in the literature:
 - “customer integration” and “customer relationship management” in marketing,
 - “co-production” and “service encounter management” in service operations research, and
 - “citizen participation” in the public policy literature.

CUSTOMER INVOLVEMENT

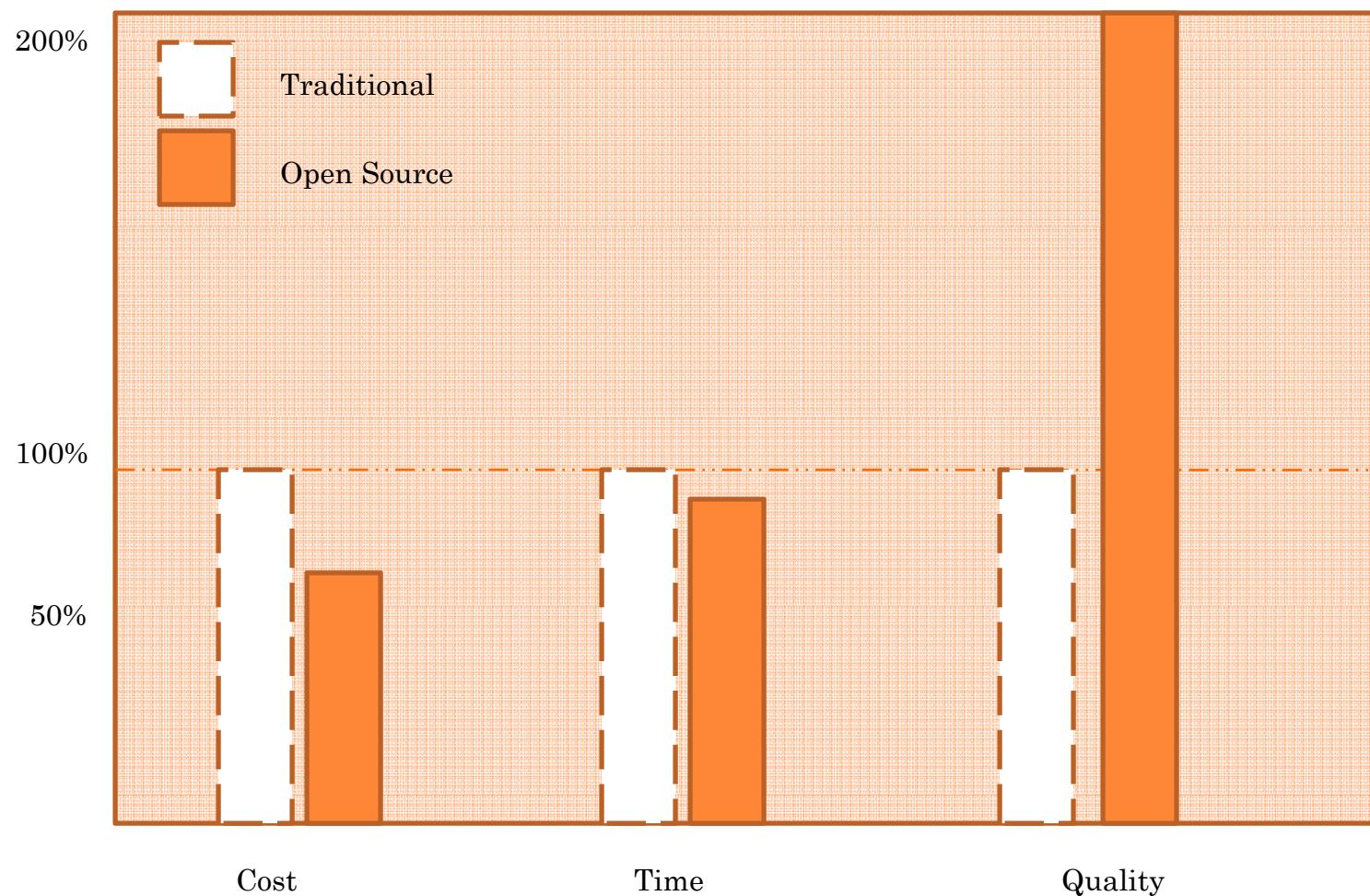
- “customer integration strategy” within the context of marketing strategies in industrial markets:
 - defining it as the ability to adapt to specific customer needs to increase business benefits
- In the software industry open source software has emerged as:
 - an important implementation of CUI economics.



MAJOR, GENERIC CUI BUSINESS VALUE CATEGORIES

Cost	Revenue
Customer or user operates business activities	
<ul style="list-style-type: none"> • Company saves employee time and expense • Likely higher fixed cost for IS that can be operated by many customers instead of a few employee only 	<ul style="list-style-type: none"> • Good can be purchased anytime and from anywhere <ul style="list-style-type: none"> • Higher quality, better product fit • Better customer data
Higher quality, better fit	
<ul style="list-style-type: none"> • Less inventory in entire channel system • Less slow moving and obsolete items • Less discounts 	<ul style="list-style-type: none"> • Customer likes the feeling of being in control -> Higher customer satisfaction • Monopolistic competition pricing opportunities
Higher customer satisfaction	
<ul style="list-style-type: none"> • Lower churn saves customer acquisition cost • Positive word of mouth may save marketing expenses 	<ul style="list-style-type: none"> • Higher loyalty • Higher lifetime customer value
Better customer data or “profiles” (behavior, wants and needs)	
<ul style="list-style-type: none"> • Data mining improves accuracy of targeting customers and saves marketing and sales cost • Lower marketing research cost 	<ul style="list-style-type: none"> • Up-selling opportunities • Better next generation product • User lock-in and higher switching cost

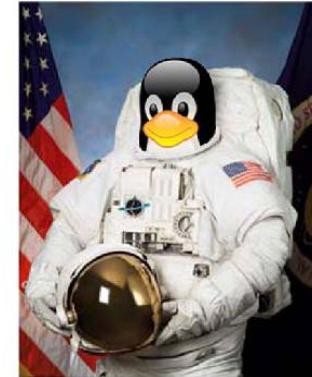
CUI BUSINESS VALUE ASSESSMENT: THE OPEN SOURCE EXAMPLE



OPEN SOURCE TIME AND COST

- In terms of cost:

- open source saves at least the profit or profit margin associated with:
 - brand name product,
 - brand name systems integration service and
 - brand name maintenance contract.
- The software business:
 - “has an exceedingly high gross [profit] margin of 90%, [...] a net profit margin of 27%.
 - This shows that its marketing and administration costs are very high, while its cost of sales and operating costs are relatively low (McClure, 2004)



OPEN SOURCE TIME AND COST

- Open source can save time:
 - because documentation is public and exposed to public scrutiny, just like the source code itself
- Customer support is not limited to a vendor's office hours or a particular maintenance subscription level
 - but open source documentation and expertise tends to be available online and anytime.



OPEN SOURCE QUALITY

- Quality can be better:
 - firstly, because of transparency of the process and
 - secondly, because of transparency of qualification and achievements of contributors
 - This mirrors a key lesson of a free market system:
 - namely that transparency tends to increase buyer value
 - Also, problems are fixed when
 - a problem exceeds users' willingness to cope and not when decided by a vendor's corporate strategy or business policy



PREVIOUS COMPARISON CHARACTERISTIC

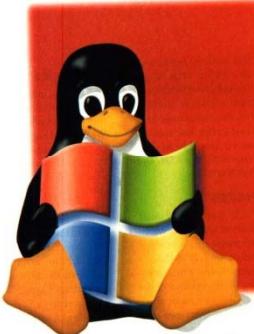
- These comparison is limited to:
 - an implementation that utilizes open source software instead of a commercial package
 - e.g., installing, configuring, integrating, testing, and maintenance
 - It does not include the writing of application source code



OUTLINE-2

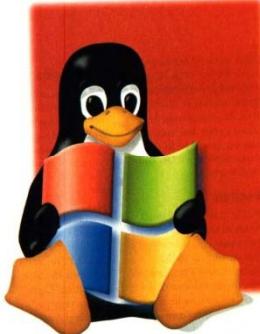
- Software Business Operations
- Software Business Entrepreneurship
- Open Source Software Initiation
- Open Source Business Models and Revenue
- Open Source Software Operations
- Commercial Open Source

COMMERCIAL OPEN SOURCE



- Commercial open source software projects are :
 - open source software projects that are owned by a single firm
 - that derives a direct and significant revenue stream from the software
- Using a commercial open source approach:
 - firms can get to market faster
 - with a superior product at lower cost than possible for traditional competitor
- The benefits of commercial open source stem from:
 - the creation of an active and engaged user community around the product
 - while at the same time preventing the emergence of competitors from that community
- In a nutshell, this community helps the company:
 - get to market faster,
 - create a superior product, and
 - sell more easily,
 - all at a lower cost than possible for traditional competitors.

COMMERCIAL OPEN SOURCE



- In exchange, the company offers:
 - a professionally developed product of compelling value to the community
 - that this community is free to use under an open source license
- *Community open source* is open source software that is owned by a community or a legal entity representing the community.
 - The community members typically don't derive direct revenues from the software
 - but subsidize it from ancillary products and service
- *Commercial open source*, in contrast, is open source software that:
 - is owned by a single legal entity with the purpose of deriving revenues from the software

HOW GENERATE REVENUE FROM OPEN SOURCE SOFTWARE

- Watson et al. distinguish five models of software production and distribution :
 - Three of these they call open source business models:
 - The “corporate distribution” model:
 - encompasses the providers of software distributions,
 - E.g. Red Hat or Spike –Source



HOW GENERATE REVENUE FROM OPEN SOURCE SOFTWARE

- Watson et al. distinguish five models of software production and distribution :
 - “Sponsored open source” :
 - is open source that does not generate revenue for the contributing companies,
 - E.g. Apache Software Foundation or Eclipse Foundation projects.
 - Finally, “second-generation open source” :
 - is open source where supporting companies generate revenue from complementary services.
 - puts all revenue generating strategies into one basket without drawing distinctions between such different models as consulting and implementation services,
 - e.g. JBoss, or license sales, e.g. MySQL.

MORE ON COMMERCIAL OPEN SOURCE

- Perhaps the clearest account of commercial open source has been provided by Michael Olson:
 - in his discussion of the “dual licensing strategy” of commercial open source firms
 - focuses on intellectual property ownership and the business strategies resulting from such ownership,
 - most notably the right to provide the product under both a (free) open source license and a (paid-for) commercial license.
- These two types of projects are distinguished by their different control and ownership structures.
 - *Community open source* is open source that is controlled by a community of stakeholders;
 - *Commercial open source* is controlled by exactly one stakeholder with the purpose of commercially exploiting it

COMMUNITY OPEN SOURCE

- Examples of community open source projects with a diverse set of stakeholders are:
 - the Linux operating system
 - the Apache web server and
 - the PostgreSQL database
 - The source code of these projects is available under one and only one license,
 - and anyone can enter the market
 - and generate revenue from the project without being disadvantaged



COMMUNITY OPEN SOURCE

- The contributors to community open source projects:
 - used to be the group of volunteer software programmers
 - who developed the open source project
 - In this case, control is determined by ownership of copyright to the code in the project and
 - related intellectual property as well as social structures such as having the commit (write) rights to the code repository



GENERATING REVENUE FROM OPEN SOURCE

- There are many ways of generating revenue from open source software, including community open source. The three dominant ones are:
 - consulting and support services around the software,
 - derivative products built on the community project, and
 - increased revenue in ancillary layers of the software stack.
- Commercial open source firms differ from traditional software vendors:
 - by not only providing the product for free as an easily installable binary
 - but also by providing it in source code form
 - By providing the source code under an open source license, such firms qualify as open source firms
 - However, because these firms own the copyright to the product,
 - they are not constrained to only one license
 - but rather they can relicense the software to customers as they see fit

GENERATING REVENUE FROM OPEN SOURCE

- Typically, the free open source form is provided:
 - under a reciprocal license like the GPL to drive adoption but stall possible competitors
 - Paid-for versions of the software are then provided under a commercial license like traditional software vendors do.
 - This is also known as the dual-license strategy of commercial open source



CATEGORIES OF PRODUCT AND SERVICE CUSTOMERS PAY FOR-1

- Four categories:
 - *Core product.*
 - Some customers pay for the software,
 - simply because they cannot accept the open source license. Mostly, this is for legal reasons
 - E.g. companies may pay for a commercial license to receive
 - certification or
 - indemnification or
 - to embed the software into their products
 - without having their own code touch open source code
 - *Whole product.*
 - Commercial users pay for the utility derived from using the software
 - Increasingly, the free open source product does not provide the full utility,
 - only a more comprehensive non-free commercial version does
 - To meet all requirements, commercial users have to upgrade from the free to the non-free version.



CATEGORIES OF PRODUCT AND SERVICE CUSTOMERS PAY FOR-2

- Four categories:
 - *Operational comfort.*
 - Customers also want to ensure that the software reliably fulfills its duty
 - Thus, they may be buying hot-line and technical support, subscription services to bug fixes, or real-time systems monitoring
 - There are many such non-functional requirements that companies may want to buy, many of which are specific to the software at hand
 - *Consulting services.*
 - Finally, customers may want to pay for training, documentation, and implementation services.



TERMS OF COMMERCIAL OPEN SOURCE

- Different names have been given to different aspects of commercial open source:
 - The term “dual-license strategy”:
 - refers to selling a commercial license to the project separate from the open source license
 - The term “freemium model”
 - a word play on “free” and “premium”
 - refers to withholding features from a free version and making them available only in a commercial version
 - Lampitt coined the term “open core model”
 - which combines the dual-license strategy with a freemium approach
 - Asay puts it together in what he calls a “phased approach”
 - to creating commercial open source businesses

COMMERCIAL OPEN SOURCE ISSUES

- Releasing a products source code as open source can create an engaged user community
- which can impact the various functions of the commercial open source firm in multiple positive ways
- This impact can create a significant competitive advantage over traditional (non-open-source) competitors. Thus, we Will discuss:
 - *Community management:*
 - How to create and sustain an engaged community
 - From the community then, the following benefits accrue, listed by business function:
 - *Sales:*
 - More and easier sales due to customer-side champions
 - *Marketing:*
 - More believable and cheaper marketing through engaged community
 - *Product management:*
 - Superior product thanks to broad and deep user innovation
 - *Engineering:*
 - Superior product that is developed faster thanks to fast and immediate community feedback
 - *Support:*
 - Lower support costs thanks to self-supporting user community.

DOWNSIDE OF OPEN SOURCE

- Open sourcing also has its downside:
 - for example, increased risk of getting sued for patent violations or of leaking important intellectual property.
 - catering to a non-paying user community and providing the public infrastructure for the community increases costs
 - The biggest danger, however, is:
 - that the firm's commercial product ends up competing with its own free open source project.
 - This challenges product management as will be discussed!



COMMUNITY MANAGEMENT

- Commercial open source firms are interested in creating an active and self-supporting user community
- Such a user community is key to achieving the desired business benefits
- Commercial open source firms are also interested in creating an ecosystem of developers and
 - service companies that extend the core product to increase its overall value proposition
- The main problem with seeding and growing a user community is:
 - the support cost!

COMMUNITY MANAGEMENT

- With closed source software, only the firm developing the software can provide the support
- With a rapidly growing user base, the support cost can quickly outgrow any existing revenue or cash reserves
- In *commercial open source*, almost all of the core product development work:
 - is carried out by the commercial firm,
 - with occasional contributions from the community

COMMUNITY MANAGEMENT

- Commercial open source firms address this problem by:
 - leading the community to become self-supporting.
 - For this, they provide:
 - not only an easily available product,
 - they also provide the source code to the product
 - under an open source license
 - From a user perspective, this has the following benefits:
 - *Free use:*
 - Providing the product under an open source license grants free irrevocable usage rights;
 - thus, users do not have to worry about having to pay down the road if they don't want to

COMMUNITY MANAGEMENT

- Commercial open source firms address this problem by:
 - From a user perspective, this has the following benefits:
 - *No lock-in.*
 - Because the source code is available under an open source license,
 - users can become independent of the commercial firm and hence (sometimes naively) think are not locked into the firms future decisions
 - *Self-support:*
 - Because the source code is open source,
 - users can solve their problems themselves without having to resort to asking the firm,
 - which might not want to provide that support to non-paying users in the first place

COMMUNITY MANAGEMENT

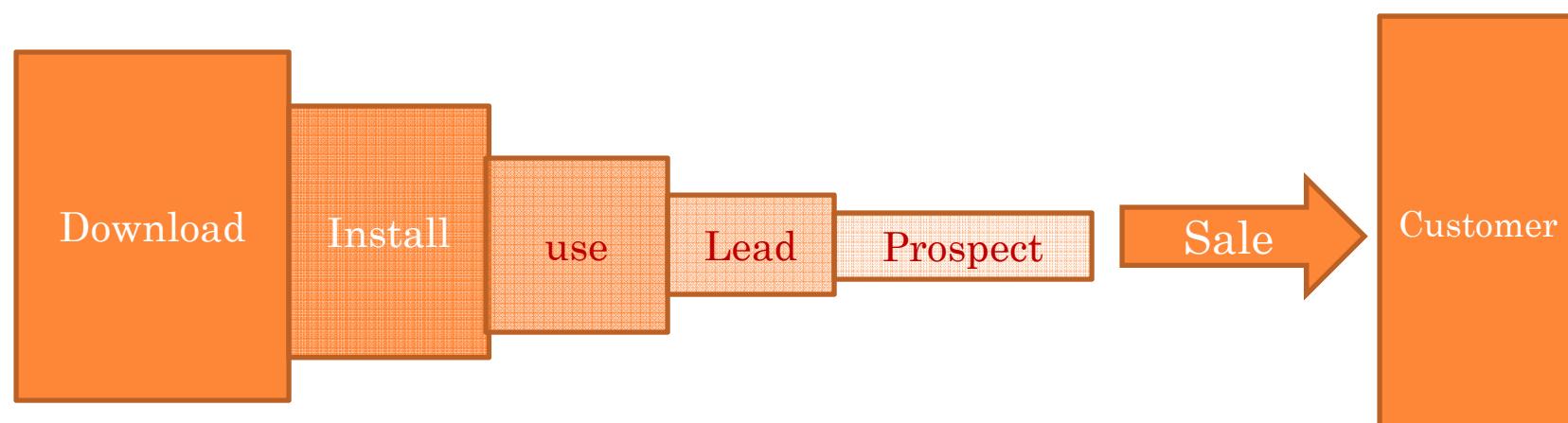
- From the firm's perspective:
 - providing the product as open source accelerates adoption without increasing support costs
 - Specifically, it reduces hurdles to adoption as potential users perceive no or little lock-in, and
 - it makes it possible that the community becomes self-supporting once it reaches critical mass.
- On the most basic level:
 - communities need a place to gather, and
 - they need tools of communication

COMMUNITY MANAGEMENT

- For this reason, most commercial open source firms host a software forge with:
 - integrated or ancillary tools like:
 - wikis,
 - forums, and
 - mailing lists.
- Much of the general advice on community building on the web applies,
 - like aiding the construction of explicit social structures and rewarding members for good behavior



COMMERCIAL OPEN SOURCE SALES FUNNEL



305

HOW LEADS EMERGE IN COMMERCIAL OPEN SOURCE

- Because the open source product is available for free:
 - potential customers can download,
 - install, and use the product
 - without ever getting in touch with the commercial firm behind the product
- At the same time, the firm can:
 - track via (typically voluntary) download registration and
 - community forum activities who is actually using the product
 - Some products also provide usage information back to the firm

HOW LEADS EMERGE IN COMMERCIAL OPEN SOURCE

- A lead analysis can then:
 - determine which of these users might be potential customers
 - More often than not, however, the firm will wait until a non-paying user steps forward and asks for a sales contact
 - to purchase any of the services outlined in the revenue generation section
- Thus, leads emerge from the existing user community,
 - either voluntarily or by analysis.
 - the commercial firm can still engage in a traditional sales cycle with non-using prospects as well

SALES AND MARKETING

- In a traditional setting, a software firm's product is unknown to the potential customer except through marketing material
- In the commercial open source setting:
 - the potential customer is sometimes already using the product and
 - hence is familiar with it
- Thus, from the buyer's perspective:
 - the open source project has significantly less risk associated with it
- In this situation, there is likely to be an inside champion
 - in the buyer's organization who downloaded and installed the product and is using it
- These factors make a sale
 - significantly easier than possible if the software firm had no prior relationship with the buyer



SALES AND MARKETING

- One role of the community is:
 - to support the potential buyer during the lead generation phase
- For economic reasons, the commercial firm cannot provide this support on a broad scale,
 - since only a small and hard-to-identify percentage of users might actually turn into customers
- According to Taylor, conversion rates of 0.5-2% are common for commercial open source firms
- Since the user is not paying at this stage, voluntary community support is typically acceptable.
- As soon as the user is converted into a paying customer, professional support becomes available.



MARKETING AND SALES

- What is new is that:
 - an engaged user community aids these marketing efforts
 - More specifically, the community makes marketing more effective and cheaper than possible without this support.
- Marketing is more effective because non-paying users are:
 - credible sources of good testimonials
- Good product and the positive engagement in the community:
 - Makes users evangelize and market the product themselves
 - without much support necessary from the commercial firm
- Free marketing can significantly reduce the marketing cost of a software firm,
 - and hence create a competitive advantage over a competing traditional firm



MARKETING AND SALES

- According to Augustin, the ratio of sales and marketing (S&M) expenses to research and development (R&D) expenses:
 - in traditional software firms is 2.3 (and sometimes much higher),
 - while it can be much lower for commercial open source firms
 - E.g. In the CRM space:
 - the S&M / R&D ratio of non-open-source firm Salesforce is 6,
 - while Augustin estimates the S&M / R&D ratio of a hypothetical open source CRM vendor to be 0.6,
 - suggesting significant savings in sales and marketing expenses
 - From a startup perspective, such a reduced cash burn rate:
 - increases the likelihood of survival for the commercial open source firm over the traditional firm



PRODUCT MANAGEMENT



- Von Hippel has shown how user innovation can be a significant source of product innovation for any commercial firm
- By providing the source code, firms encourage volunteers:
 - to innovate and contribute to the product for free
- As mentioned, no such contributions will be accepted unless the rights are transferred to the commercial firm
- Nevertheless, such user innovation can significantly improve the product,
 - and if only through ideas rather than code
- An engaged community actively discusses:
 - strengths and weaknesses as well as future prospects of the open source product

PRODUCT MANAGEMENT



- Almost every commercial open source software firm provides:
 - the means to such discussions in the form of
 - mailing lists, forums, and wikis on a company-run software platform
- Thus, product managers can easily observe and engage with:
 - the community and discuss current and future features
- This in turn brings product managers:
 - close to users and customers,
 - aiding the product management process,
 - E.g. by helping feature definition and creation of a product roadmap

PRODUCT MANAGEMENT



- In commercial open source:
 - this community does not only include current customers
 - but also current non-paying users and
 - possibly even researchers and students.
- Thus, compared with a traditional community of customers,
 - the breadth of perspectives in such discussions is much higher
 - This breadth of perspective in turns helps product managers understand new features and issues
 - that have kept non-users from becoming users as well as existing users from converting to customers.
- Many commercial open source firms distinguish between:
 - a free community version of the product and
 - a paid-for enterprise edition

PRODUCT MANAGEMENT



- Product management faces the challenge of:
 - motivating enterprises to purchase a commercial license
 - without annoying the non-paying community by withholding important features
- Smart product managers address this problem by:
 - determining which enterprise features are irrelevant to the open source community
 - and by taking a time-phased approach to making features available that are needed by both communities
- Product management benefits greatly from:
 - the immediate connection with the community,
 - which provides ideas and feedback and
 - keeps the product focused on its needs.
- Thus, the community helps the firm create a superior product

ENGINEERING



- *Engineering* Obviously, volunteer contributions can speed up development
- Also, an engaged technical community represents:
 - a potent pool of possible future employees
 - that proved themselves before being hired,
 - taking risk out of the hiring process
- More importantly, however, and similar to product management:
 - are the benefits of direct and immediate feedback from the community
- A commercial open source company is likely to provide:
 - the latest release, sometimes a daily release,
 - to the community, including potential bugs
- An engaged (and fearless) community picks up the latest release
 - and provides feedback to the company about bugs and issues they found,
 - sometimes together with a bug fix

ENGINEERING



- While such community behavior may appear as counterintuitive,
 - it is nevertheless what practitioners experience
- The distinction between an experimental community edition and slower-paced
 - but more stable enterprise edition in turn
 - lets the commercial open source firm sell operational comfort,
 - that is, the stable enterprise edition, more easily
 - Still, engineering management may not want these two versions to become too different from each other
 - to avoid (re-)integration problems with outside contributions
 - as well as unnecessarily redundant development on both versions

SUPPORT



- *Support* An engaged community supports itself by and large
- Users who are not customers:
 - typically don't expect professional support from the commercial firm and
 - are willing to utilize (and contribute to) community support
- The commercial firm needs to aid in the support,
 - but does not have to perform the bulk of the work
- It would be prohibitively expensive for the commercial firm to provide support to all users,
 - including those that don't pay
 - Thus, a self-supporting community is necessary to grow a large (non-paying) user base
 - that might be converted into paying customers later

SUPPORT



- Paying customers can then receive full support from the commercial firm
 - as part of their maintenance contracts
- The self-support activities of the community
 - benefit the support activities of the commercial firm as well, reducing its cost
- Specifically, engaged communities frequently develop and manage their own documentation,
 - or at least contribute to and expand company documentation
- User-maintained wikis and knowledge bases have become common
- Thanks to the power of Internet search, many users, including paying customers,
 - find it easier and faster to browse for problem solutions
 - before turning to paid support in the form of phone calls or emails
 - Thus, the community takes some of the support burden of the commercial firms shoulders, reducing the overall support expenses

CONCLUSION OF COMMERCIAL OPEN SOURCE

- Open source is changing:
 - how software is built and how money is made
- Industry analysts predict that by 2012 more than half of all open source revenue:
 - will accrue to single-vendor dominated open source projects,
 - called commercial open source
- From this user community, many benefits accrue, touching almost every business function of the firm:
 - Sales are eased and increased through inside champions and reduced customer risk,
 - marketing becomes more effective through better testimonials and active community support,
 - product management more easily meets customer needs and benefits from user innovation,
 - engineering creates a superior product faster and cheaper, and support costs are reduced
- Thus, first order of business for a commercial open source firm is:
 - to create and sustain this community,
 - a business function frequently non-existent or neglected in traditional software firms



REFERENCES

- [1] H.B.Kittlaus, P.N.Clough, “Software Product Management and Pricing key success factor for software organizations”, published by Springer, 2009.
- [2] M.Casumano, The Business of Software: What Every Manager, Programmer, and Entrepreneur Must Know to Thrive and Survive in Good Times and Bad, published by Free Press, 2004.
- [3] K.St.Amant, B.Still, “Handbook of Research on Open Source Software: Technological, Economic, and Social Perspectives”, Published by Information Science Reference, 2007.
- [4] D.Riehle, “The Commercial Open Source Business Model”, *Proceedings of the Fifteenth Americas Conference on Information Systems, 2009*.
- [5] What is Open Source Business Model,
<http://www.extropia.com/tutorials/misc/opensourcebiz.html>, visited on 2009.
- [6] S.Cohen, “Open Source: The Model Is Broken”,
http://www.businessweek.com/technology/content/nov2008/tc20081130_276152.htm, 2008
- [7] Y.Dehdashti, M.Hejazinia, E.Mehregan, “Survey over software industry”, Sharif University of Technology, 2009.