Ohjelmistoliiketoiminta nyt ja tulevaisuudessa –
teknologian ja liiketoiminnan muutokset ja mahdollisuudet

Jyrki Kontio, Ph.D.

jyrki.kontio@iki.fi
http://www.jyrkikontio.fi/

jyrki.kontio@rdware.com
http://www.rdware.com/

Presenter Background

- Principal Consultant and Founder of R & D-Ware Oy
- Board member at
  - QPR Software Oyj
  - Webropol Oy
  - Tietotekniikan Liitto
  - Ohjelmistoyrittäjät ry
- Professor of Software Product Business, TKK, 2002 – 07
- Nokia, 1986 – 2002
  - Knowledge-based systems research and consulting at Research Center (1986-92)
  - Manager of the software engineering research group at Research Center (1992-94)
  - Quality manager at a business unit (1997-99)
  - Senior manager at Nokia Networks: process management (1999-2000)
  - Principal Scientist at Nokia Research Center, software capability (2001-02)
- Other experience
  - Acting (part-time) professor of Software Engineering at Helsinki University of Technology (1997-2000)
  - Senior researcher at University of Maryland in professor Basili's research group (1994-96)
  - Software development and management in software houses and corporations (1982-1986)

Why Software Business?

Professor Emeritus David G. Messerschmitt
University of California at Berkeley

"I co-authored 'Software Ecosystem' because software is so unique among industries with respect to not only its technology, but also its business and economics characteristics. I am convinced that software is a forerunner in this respect: More and more business in developed countries, as it moves toward a service orientation, will resemble software business. Thus, if you want to be in on the forefront of business challenges, software is a great place to start.“

What is Software?

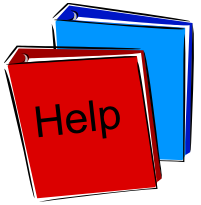
Specifications and design documentation

Source code

Customer

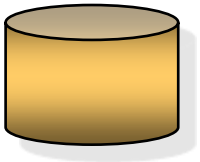
Delivery

Executable code



Manuals and guidelines

Service



Data and databases

Software exists because you plan to make money with it: the business model is an essential "component" of software

Volendam Manifesto on Software Business – Expert Panel

Professor **Sjaak Brinkkemper**, Utrecht University

- Professor of product software, meeting organizer

Professor **Anthony Finkelstein**, University College London:

- A leading software engineering scholar in Europe, ICSE-04 general chair

Professor **Alan Hevner**, National Science Foundation and U of South Florida:

- Program coordinator at NSF, published a highly cited paper ” Design Science Research in Information Systems” in MISQ

Professor **Jyrki Kontio**, Helsinki University of Technology

- Professor of Software Product Business, Head of the Software Business Laboratory

Professor **Alan MacCormack**, Harvard Business School:

- A leading U.S. scholar in software business and software architecting from business perspective

Professor **Tony Wasserman**, Carnegie Mellon West University:

- Experienced software entrepreneur, founder and president of IDE (Software through Pictures, reached Top 500 Inc. list)

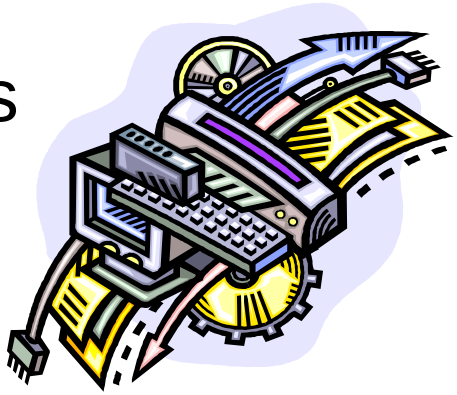
Definitions

- **Software Ecosystem** is a specific perspective to a subset of an economy where software constitutes a substantial part of the transactions between agents on the marketplace.
- **Software Business** refers to transactions between agents, trading software-based assets.
  - *Agents* are organizations or individuals;
  - *Assets* include financial assets, buildings, other resources, capabilities, and, of course, software
  - *Transactions* include monetary transactions, transfer of goods, provision of services, and, of course, delivery and deployment of ***software***
- **Software Business Research Field** studies how these assets are created to create value, how agents interact and what kind of transactions take place, and how technologies are used to support this business.
- We study issues related to bringing software assets to the market. Specifically, we study how these assets are developed to create value, how agents interact, and what kind of transactions take place, and how technologies are used to support this business.

Characteristics of Software

- Complexity
- Conformity
- Changeability
- Invisibility

- Configurability
- Human development process
- Technological change rate
- Digital good
- Networked good
- Experience good
- Need for services



Brooks: No Silver Bullet

Complexity

- Each unit is unique and interacts with many other units
- Abstraction (easily) hides complexity
- Yet complexity is an essential feature of software

Conformity

- There are no natural laws governing software
- Human judgment has determined many of the “constants” in software
  - many of them are not compatible
- Software must conform to other disciplines

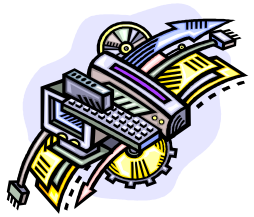
Changeability

- Software is tied to other “systems” that evolve:
  - user needs
  - organization
  - laws, society
  - hardware platforms
  - other software systems
- Change is a given environmental factor

Invisibility

- Can’t touch, feel, or smell
- Models (abstractions) are always partial and multidimensional
- It is difficult to understand what software is

Other Technical Characteristics of Software



Configurability

- Software can be configured near or at runtime into different configurations
- Mass customization at point of sale possible

Human development process

- Software engineering very people dependent:
  - Human creativity
  - Subjectivity of specification and design, and
  - Dependence on teamwork

Technological change rate

- No other technology evolves so rapidly
  - ”Programming languages change in less than 10 years”
  - ”Application platforms change every 5 years”
  - ”Development paradigms change every 10 years”
  - ”Development environments are updated annually”

Digital good

- Software is entirely digital
- Can be copied error-free
- Can be digitally transmitted

Networked good

- Software is **technically** linked to many other systems
- Software creates many **social** interactions and dependencies

Experience good

- The quality and utility can be assessed (only) by using the software
- Value of software is realized through its use

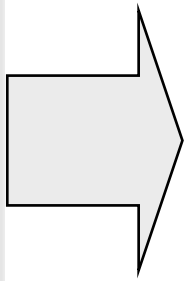
Need for services

- Deployment of software often requires integration and training services
- Use of software requires support services (maintenance, training, helpdesk, etc.)

From Technical to Business Characteristics

Characteristics of Software

- Complexity
- Conformity
- Changeability
- Invisibility
- Configurability
- Human development process
- Technological change rate
- Digital good
- Networked good
- Experience good
- Need for services

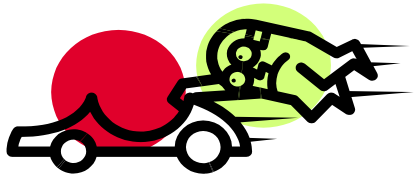


Implications to Business

- Easily mobilized
- Networked on multiple levels
- Economies of scope
- Economies of scale
- Economies of aggregation
- Instant scalability
- Large design costs
- Low reproduction costs
- Low distribution costs
- Experience good
- Lock-in
- Network effects

Software is Easily Mobilized

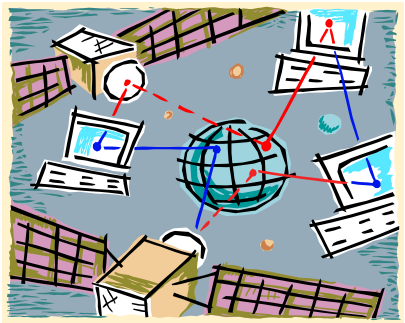
- Software can be taken to use easily, regardless of time and place
  - Downloaded on demand
  - Automatic installers
  - No storage costs
  - Instant and low cost access in many situations



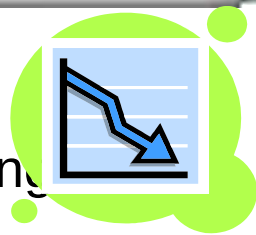
- Advances in usability have made instant use common
- Network access required
- Examples:
  - Software upgrades, e.g., anti-virus software
  - Mobile phone game downloads
  - Time-based capacity in a network

Software is Networked on Multiple Levels

- Software is technically integrated with many other systems
- Software integrates with processes and conventions in any organization
- People interact through software: social lin networks are created through software
- Examples:
  - Instant messaging applications
  - SAP
  - Music sharing software



Software has Economies of Scale



- Definition:
  - Economies of scale exist when the cost of production/manufacturing decreases when large numbers of the good are produced
- Once developed, software has tremendous economies of scale
- Integration and training costs reduce immediate economies of scale
  - Standardizing (”productizing”) services and integration will improve economies of scale
- Examples:
  - Microsoft products, any packaged, high volume software

Software has Economies of Scope

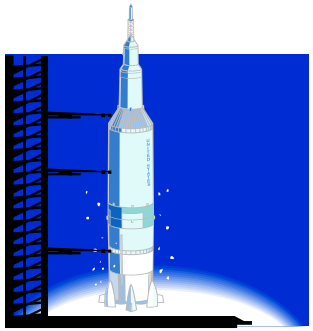
- Definition:
  - “Fewer inputs, such as effort and time, are needed to produce a greater variety of outputs.
  - Greater business value is achieved by jointly producing different outputs. Producing each output independently fails to leverage commonalities that affect costs.” (www.sei.cmu.edu/productlines/glossary.html)
- Good architecting and business planning increase potential for economies of scope
- Examples:
  - IT companies offering products for the telecom sector
  - Google offering its generic search engine also for company intranets

Software has Economies of Aggregation

- Definition:
  - Aggregation means bundling of offering so that more functionality is delivered to customer
- Large firms with a large product portfolio can bundle their offering to create offerings that competitive both in terms of functionality and cost (per function)
- Leads to market dominance and ”winner takes it all”
- Examples:
  - Microsoft office: includes email, browser, all office tools

Software has Instant Business Scalability

- Software has nearly instant business scalability: possibility to deliver large quantities with relatively small change in the use of resources
- In practice, scalability is limited through:
  - Human sales efforts
  - Logistics (traditional delivery)
  - Invoicing
  - Support requests



- Examples:
  - Internet allows easy and fast distribution
  - Netscape invested in high-volume business applications early
  - Amazon.com built up scalability early

Software has High Design Costs

- Development of the first version of software is costly and time-consuming
- Business case for software is largely dependant on the volume of use
- Examples:
  - Software customization
  - New release development

Software has Low Reproduction Costs

- Software has low variable costs: a new copy of software can be reproduced at close to zero marginal costs
- Reproduction costs are raised by:
  - Sales efforts
  - Logistics (traditional delivery)
  - Invoicing
  - Associated hardware or other physical material (e.g., manuals)
- Examples:
  - SAP has high marginal costs
  - Downloadable software has nearly zero marginal costs

Software can have Low Distribution Costs

- Being digital, software can be delivered electronically
- Most of the bandwidth can be obtained at close to zero marginal costs
- Distribution costs raised when
  - Delivery through physical logistics chain (VARs, shrink-wrap software)
- Distribution costs are paid by the end cu
- Examples:
  - NET is effectively a distributor for some SW
  - Software downloads

Software is an Experience Good

- **Experience good**: product characteristics are difficult to observe in advance, but these characteristics can be ascertained upon consumption or use
- Any description or metric about software is bound to be an abstract summary of software
- Examples:
  - SAP
  - Network management software
  - iTunes software
- 1

Software has Strong Lock-in Effects

- Definition:
  - Initial sales or ownership of products create an investment for the customer that makes him reluctant to change products or vendors
- Software lock-in occurs, e.g., through
  - Customization of software
  - Integration of software with other systems
  - Learning curve costs
  - Customer specific knowledge in services
- Examples:
  - Mobile phone
  - SAP
  - Microsoft Windows operating system

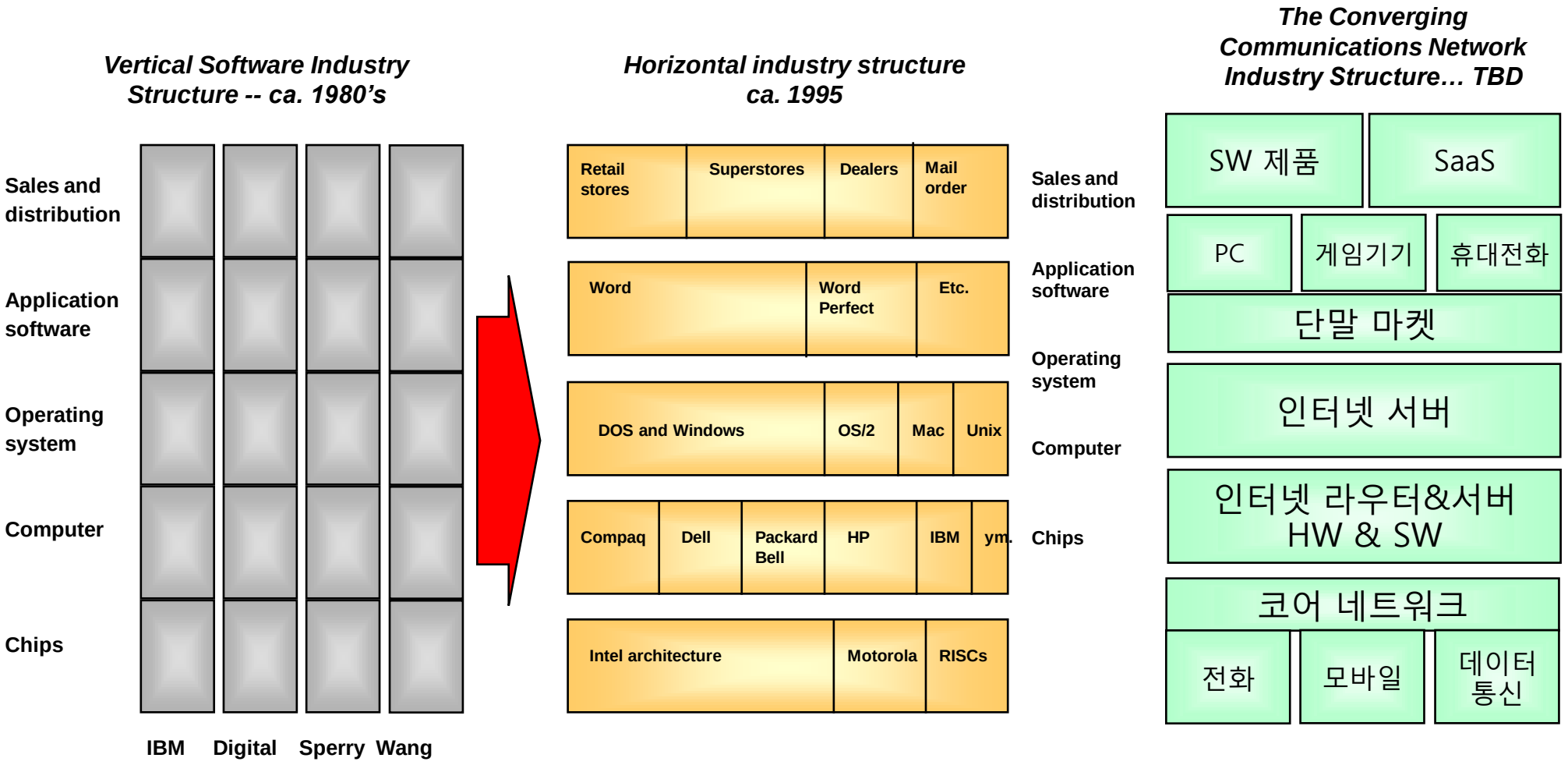
Software has Strong Network Externalities

- Definition:
  - "effects on a user of a product or service of others using the same or compatible products or services" (http://www.answers.com/)
  - "a cost or benefit that arises from production and falls on someone else than the producer" (Bowman & Ambrosini, 2000)
- Examples
  - Positive network externalities:
    - Consultant service available
    - Fax machine, MS-Office
    - GSM standard
  - Negative network externalities:
    - Other users on a shared Internet connection

Trends affecting Software Business

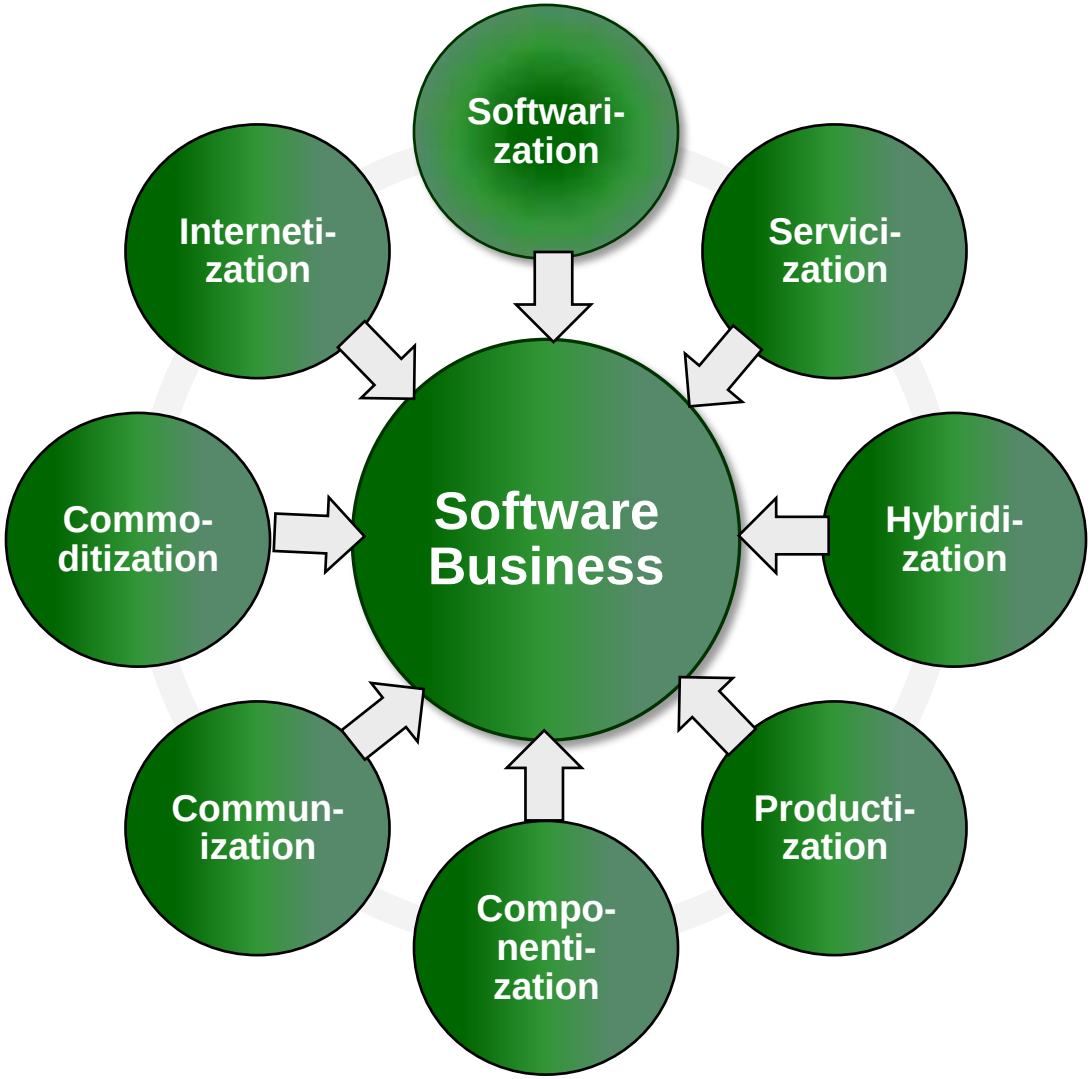
- **Softwarization** – “ohjelmistoituminen”
  - Services and functionality in traditional products are offered through software
- **Servicization** – “palvelullistuminen”
  - Software functionality is increasingly offered as a “service”
- **Hybridization** – ”hybridisoituminen”
  - Functionality, services and content together create value to users
- **Productization** – ”tuotteistuminen”
  - Both software and services are being ”productized” for efficiency and profitability
- **Componentization** – ”komponentointuminen”
  - Components are increasingly used as building blocks of systems
- **Communization** – ”yhteisöllistyminen”
  - User communities are involved in the development of software
- **Commoditization** – “tavallistuminen”
  - Hardware and basic software functionality are becoming standard and available (cheap or free)
- **Internetization** – ”internetistyminen”
  - Internet is the platform – and it’s mobile

Example of an Industry Change

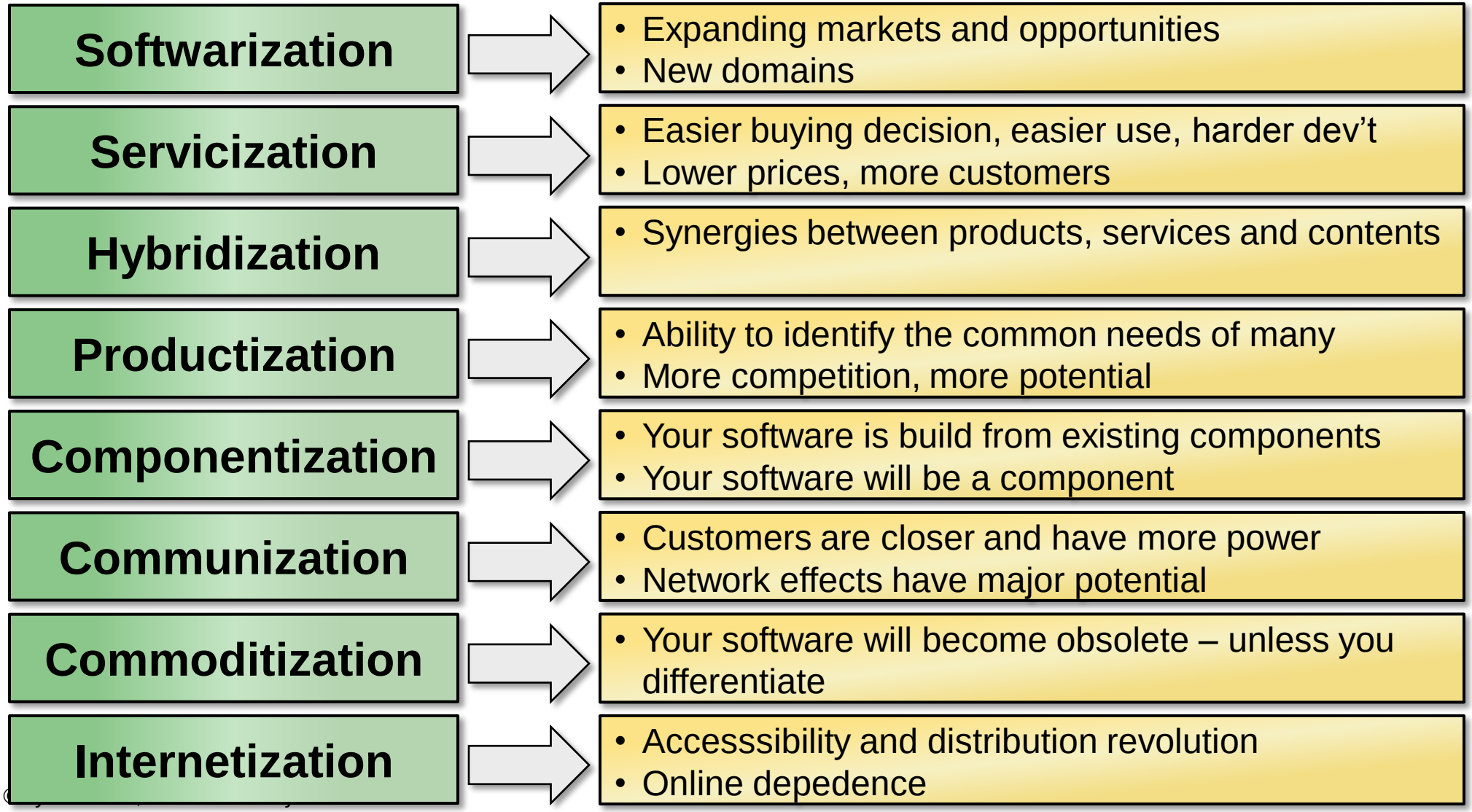


Andy Grove, 1996

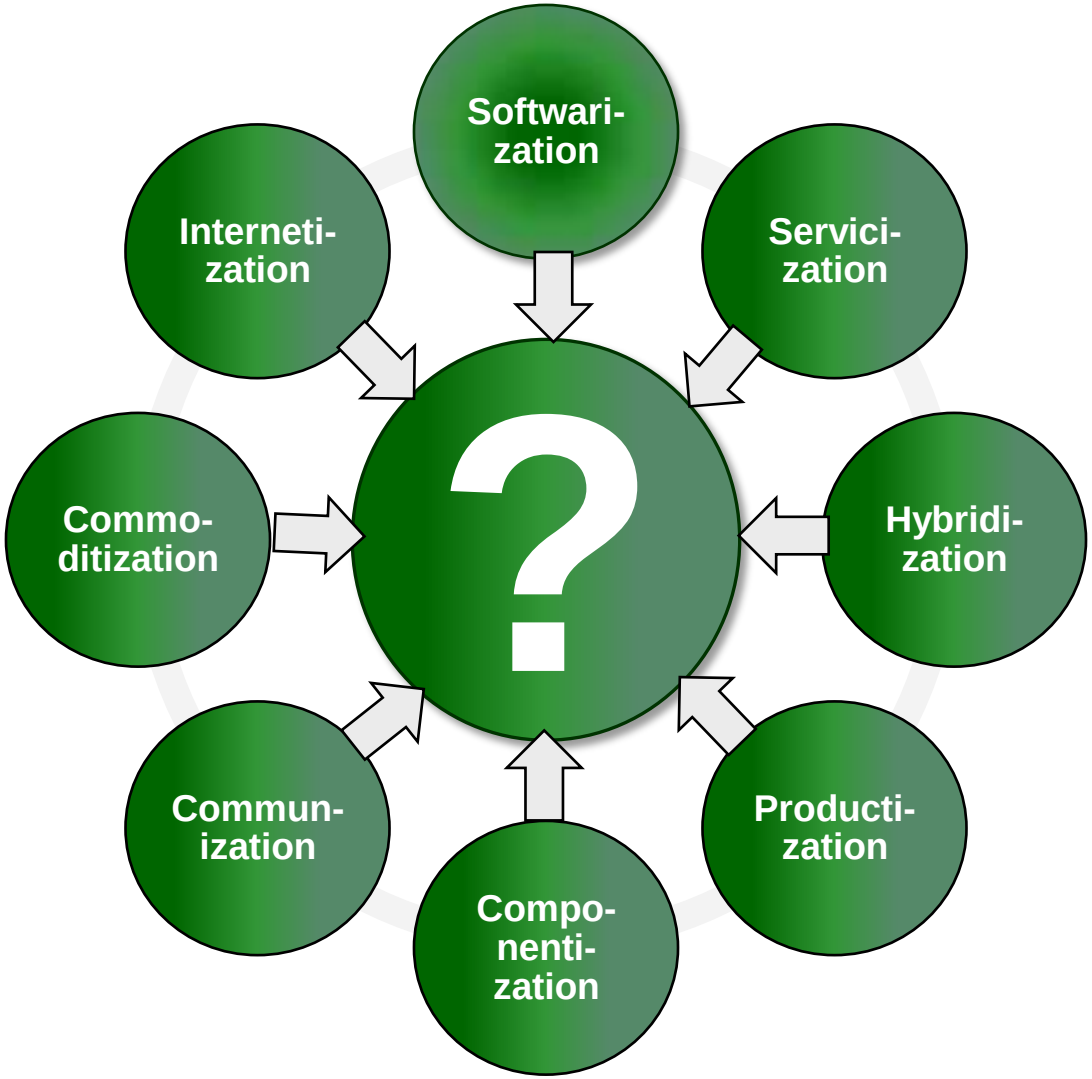
”Kokonaisuus on enemmän kuin osien summa”



Implications of Trends



”Kokonaisuus on enemmän kuin osien summa”



Uusi ”systeemityö”

- Business on osa systeemiä
  - ➔ **suunnittele business, älä sovellusta**
- Elämme muutosta
  - ➔ **kokeilu ja jousto ovat välttämättömiä**
- Varaudu koko elinkaareen
  - ➔ **onnistunut tuote elää pitkään**
- Oikeat kyvykkyydet ovat tärkeämpiä kuin oikea tuote juuri nyt
  - ➔ **tunnista organisaatiollesi tärkeät kyvykkyydet ja kehitä niitä**