Systems Integration

4 - Builder-Architected Systems

Pere Palà

iTIC http://itic.cat

v1.2 September 2016

Source: A significant part is from Mark W. Maier and Eberhardt Rechtin's *The Art of Systems Engineering 3rd Ed*

Form-First vs. Function-First

Form-First

- Begins with a builder-conceived architecture (not client-accepted purposes)
- Architects are members of the technical staff of the company

Incremental Developments

- New product as a variation of existing one
- If original is ok, variations/extensions are low-risk
- Key: Identify the critical architectural features to keep advantage in the market

Judgement of value

- Function-First (purpose driven): Value is stated before
- Form-First (form driven): Value is stated after

Innovative Developments

New markets for existing products

- New product requires partial re-architecture
- Try to preserve the existing product line
- Example: basic product with options

New products for new markets

- Higher risk: changes in kind rather than in degree
- Often triggered by radically new technologies (jet engines, new materials, microprocessors, lasers ...)
- Greatest risk is timing:
  - Too early
  - Too late

Technological Substitutions within Existing Systems

Much more complexity than foreseen

- It is not just a matter of substituting components from old to new technology
  - Factory workers with machines
  - Vacuum tubes with transistors
  - Large inventories with just-in-time
- Evolution is a combination of elements
  - e-global community: packet switching, fiber-optic lines, www, e-commerce
- Manufacturing management
- Financing
- Regulations
- Environmental impact

Uncertainties and Risk

Uncertainty of end purpose

- Errors in decisions: design, development, production
- Solutions looking for a problem. Perhaps the wrong problem

Reducing risks

- Build and maintain options
  - Options to stop at known points
  - Options for additions
  - Options to isolate troubled sections
- Use open architectures. You will need them once the market starts to respond
- Pause and reflect

Architecture and Competition

No competition in classical architecting! Competition strategies are

- Disrupt and dominate
  - Great quality. Imagination. Options
  - Put barriers to competition who may have an *agile response*
- Agile response
  - Exploits (not disrupts) the flux of the market
  - Quick **and** effective response
  - Fast architecting
  - Organizational structure has to support agility
- Attrition
  - Low-cost capital, low-wage labor, large distribution channels
  - For mature markets
  - Fails if there is a larger competitor...
  - ...or there are quick changes

Reducing Risk via Intermediate Goals

- Plan intermediate goals
  - Simulators, prototypes
  - Build partial systems (demonstrators or models): assess if customer's view matches architect's one
- They have to be well architected (and designed and built) too!
  - A failure can ruin the program
  - A success does not imply success of the whole system
- *Do the hard parts first*. If not, high risk is maintained till the end

What next

- First risk: lack of a successor
  - Start-up companies may not have enough resources to support more than one R&D project
  - A good product will attract more competition
  - The original product becomes a “commodity”
- Well established companies unable to win a contract for the follow-on
  - The positive aspects of the original system: architecture plus organization may to be too slow to react when technologies change or market needs change
  - The present system is too engraved in the thinking. There do not appear new options

Keep Control of the Critical Features

- Keep ownership of the basic features (disrupt and *dominate*)
  - Operating systems, interface characteristics, communication protocols, patents, exclusive agreements with suppliers or distributors
  - Good products are required but they must establish a new standard
  - Success invites competition: make competitors dependent through licensing
  - *Successful architectures are proprietary, but open*
- Personal computers
  - Microsoft licensed the operating system. Apple did not
  - Expanding of clones was successful for Microsoft
  - But too open is also a problem: IBM made the PC too open!
  - Microsoft and Intel made balance between open and protection
- Too open: loose competitive advantage. Too closed: unable to create synergies

Abandon Obsolete Architectures

- Well-established product-line architecture but is losing out
- Competitor provides a better way to do things
  - Typewriters replaced by PCs
  - Energy costs rise
  - Worksations replaced by PCs
- Cannibalize the old architecture, including the organization matched with it
  - Xerox example: From copiers to “the document company”

Creating Innovative Teams. The Four Temperaments

- SJ - The Guardians. Security Seeking
  - ESTJ - "The Supervisors"
  - ISTJ - "The Inspectors"
  - ESFJ - "The Providers"
  - ISFJ - "The Protectors"
- SP - The Artisans. Sensation Seeking
  - ESTP - "The Promoters"
  - ISTP - "The Crafters"
  - ESFP - "The Performers"
  - ISFP - "The Composers"
- NT - The Rationals. Knowledge Seeking
  - ENTJ - "The Fieldmarshals"
  - INTJ - "The Masterminds"
  - ENTP - "The Inventors"
  - INTP - "The Architects"
- NF - The Idealists. Identity Seeking
  - ENFJ - "The Teachers"
  - INFJ - "The Counselors"
  - ENFP - "The Champions"
  - INFP - "The Healers"

Extraversion/**I**ntroversion. **S**ensing/i**N**tuition. **T**hinking/**F**eeling. **P**erception/**J**udgment.

Creating Innovative Teams

Personalities

- NT personality preferred
- Systems Architect INTP
- Include an ENTP, ENTJ

The product of a single mind

- Single mind for modest-sized projects. Multidisciplinary team
- How are decisions taken? Diversity of view but unity of decision
- Whole team has to be a single mind regarding the architecture

Innovative teams

- Informal creativity, easy interpersonal relationships, trust. Believe in chief
- Diversity in specialization, in style, interests

Architecting Revolutionary Systems

- All have a clearly identifiable Architect
- Not conceived by consensus of a committee

Architect and Project Manager

- Architect: leads technical part
- Project Manager: involved with short term problems

Success

- Commonly not found where originally thought
- Macintosh computer: desktop publishing
- The *killer app*: a system usage so valuable that it, by itself, drives the dissemination of the system
- Search for the killer app!

Heuristics for Architecting Technology-Driven Systems

- An insight is worth a thousand market surveys
- Success is determined by the customer, not by the architect
- In architecting a new program, all the serious mistakes are made in the first day
- The most dangerous assumptions are the unstated ones
- The choice between products may well depend upon which set of drawbacks the users can handle best
- As time to delivery decreases, the threat to user utility increases
- If you think your design is perfect, it is only because you have not shown it to someone else
- If you do not understand the existing system, you cannot be sure you are building a better one
- Do the hard parts first

Heuristics for Architecting Technology-Driven Systems/2

- Watch out for domain-specific systems. They may become traps instead of useful system niches, especially in an era of rapidly developing technology
- The team that created and built a presently successful product is often the best one for its evolution –but seldom for creating its replacement. (It may be locked into unstated assumptions that no longer hold)
- Good products are not enough. (Their features have to be owned)
- Implementations matter. (They help establish architectural control)
- Successful architectures are proprietary but open. (Maintain control over the key standards, protocols, etc., that characterize them but make them available to others who can expand the market to everyone's gain)
- Use open architectures. You will need them once the market starts to respond