Systems Integration

Design Progression

Pere Palà

iTIC http://itic.cat

v1.0 December 2013

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

Introduction

Process of architecting

- Systems are diverse. No dogmatic approach
- Concepts ofr architecting activities
- Design process is eclectic. Organization possible
- Key concept: Refinement
- System models: from objective to implementation
- Difference with conventional engineering: parallel development of problem and solution
- Problem is not assumed to be fixed

Design Progression

- Common pattern: progressive refinement
- Way to organize the transition
  - From ill-structured, chaotic and heuristic process at the beginning
  - To rigorous engineering and certification needed later
- Can be thought of a stepwise reduction of abstraction
- Accompanied by an increase in volume of information about the system
- Episodes of abstraction reduction with episodes of reflection and purpose expansion

Introduction by Examples

- Civil architect developing a building
  - First drawings: rough floor plants and external renderings
  - Construction details come later
- Stepwise refinement in programming
  - Controlling routine first
  - When high complexity is found: ignore, give a name, become a subroutine
  - Stubbed subroutines
- Both examples show
  - Progression of modeling
  - Strategy: ordering of decisions
- Both should create distinct alternative designs
  - Estimate cost, aesthetic opinion...
  - Code size, execution speed, review functionality...

Evaluation Criteria and Heuristic Refinement

- Desirable progression for evaluation: from general to system-specific to quantitative
- Desirable progression for heuristics: from descriptive and prescriptive qualitatives to domain-specific quantitatives and rational metrics
  In partitioning, choose the elements so that they are as independent as possible –that is, elements with low external complexity and high internal cohesion
- Heuristic that is independent of domain. Guidance is non-specific. Independence? Complexity?
- Moving to a restricted domain: computer-based systems: *Module fan-in should be maximized. Module fan-out should not exceed* 7 ± 2

Evaluation Criteria and Heuristic Refinement /2

- Should be further refined into quantitative design quality metrics
  Compute a complexity score summing: 1 point for each line of code. 2 points for each decision point. 5 points for each external routine call. 2 points for each write to a module variable. 10 points for each write to a global variable.
- Other metrics exist. Cyclomatic complexity

Progression in Corporations

- A product. An engineering challenge
- An element of value. A source of profit
- Acquires assumption of permanence. Established corporation encompassing the system, its ongoing development and its support

Concurrent Progressions

- Risk management
  - Early risk management: heuristic with mix of rational methods
  - Prototypes, experiments: risk management mixed with interpretation of results. Estimates are replaced by information
  - After system construction: risk management is post-incident diagnostics
- Cost estimates
  - Early stages: high need for estimate, low information available. Uncertainties are still unresolved
  - As development proceeds: design and plans are more concrete and costs already have been incurred (no estimates but reality)
  - Process of decreasing need and increasing information available
- Reliability
  - Customer's desires known, but performance unknown
  - Reliability estimates become known as design progresses to lower levels
  - Known when measured in the field

Architecting is Episodic

- Not a monotonic process

Design Concepts

- Architecting is a mix of rational and heuristic engineering
- Architecting revolves around models. Scoping, synthesis and certification
- Synthesis: Creative invention
- Uncertainty is inherent in complex system design
- Continuous progression on many fronts
- Architecting is combining science with art

Scoping, Synthesis and Certification

Scoping

- Purpose expansion / contraction
- Behavioral definition / analysis
- Large scale alternative definition
- Client satisfaction and builder feasibility

Scoping, Synthesis and Certification /2

Synthesis

- Problem reformulation
- Creative invention
- Iteration
- Aggregation
  - Functional aggregation
  - Physical components to subsystems
  - Interface definition /analysis
  - Collection into decoupled threads
- Partitioning
  - Behavioral-functional decomposition
  - Physical decomposition
  - Performance model construction
  - Interface definition /analysis
  - Decomposition into threads

Scoping, Synthesis and Certification /3

Certification

- Operational walkthroughs
- Test and evaluation
- Verification
- Formal methods verification
- Failure assessment

Scoping

- Well-scoped system: desirable and feasible
- Participants form mental model of the system
- *All the really important mistakes are made the first day*
- Defer absolute decisions on scope
- *Success is defined by the beholder, not by the architect*
- *Listen closely to what the customer perceives as his requirements and have the will and ability to be responsive*
- *Ask early about how you will evaluate the success of your efforts*
- *Moving to a larger purpose widens the range of solutions*

Synthesis

- Synthesis is creation
- *Often the most striking and innovative solutions come from realizing that your concept of the problem was wrong*
- Database synchronization $\rightarrow$ buy communications capacity
- *Plan to throw one away, you will anyway*
- Innovative solutions will require to throw away early attempts
- Aggregation and partitioning
- *In partitioning, choose the elements so that they are as independent as possible –that is, elements with low external complexity and high internal cohesion*
- *Group elements that are strongly related to each other; separate elements that are unrelated*
- There are metrics for cohesion and partitioning in software

Certification

- To give assurance to the paying client that the system is fit for use
- House: visual inspection. Computer system: extensive inspection, mathematical proofs...
- Certification should not be treated separately from scoping or design. Must be inherent in the design
- *For a system to meet its acceptance criteria to the satisfaction of all parties, it must be architected, designed and built to do so –no more and no less*
- *Define how an acceptance criterion is to be certified at the same time the criterion is established*
- Defect discovery: trace to the source
- *Enumerate the defects, analyze them, trace them to the source, make corrections, keep a record of what happens afterwards and keep repeating it*
- Certification of ultraquality
- *The number of defects remaining in a system after a given level of test is proportional to the number found during that*

Process Model

Activities

- Orientation
  - Scoping / planning
- Core architecting (Aggregation / partitioning)
  - Purpose analysis (elicitation)
  - Problem structuring (synthesis)
  - Solution structuring (synthesis)
  - Harmonization (analysis)
  - Selection or abstraction (decision making)
- Architecture description
- Supporting analysis

Some Considerations

Orientation

- What sort of system does the sponsor believe will emerge?
- What is the scope of the system? Single-mission? Complex multi-mission? Collaborative system?
- What is the required technology level? Within state-of-practice? Beyond?
- What are the hard constraints (date)? Are they *hard*?
- What resources are available?
- What will be done after architecting is complete?
- Are the purposes, architecting effort and documentation required consistent with each other?
- What is the motivation of constructing the system?

Some Considerations /2

Purpose Analysis

- *Who benefits, who supplies, who pays and who loses?*

Problem Structuring

- Problem framing, expansion and contraction heuristics, use-case analysis and functional decomposition

Solution Structuring

- Products are models of the system. Block diagrams, ...

Harmonization

- Match up problem and solution
- Functional walkthroughs, performance analysis and executable simulations

Some Considerations /3

Selection or Abstraction

- Make choices. Perhaps drop the whole pursuit!
- Select the desired configuration
- Abstraction. For collaborative systems, select common things

Architecture Description

- It is the *result* of architecting work

Supporting Study

- Deep investigation of narrow areas. Key for system performance
- One architecting cycle reveals the areas requiring in-depth investigation

Decisions

- Decision theory works well in ideal situations: reliable data, cost function available...
- Elements of decision
  - Identifiy attributes contributing to client satisfaction
  - Determine a utility function
  - Include uncertainty by determining probabilities. Use client's risk aversion curve
  - Select result with highest expected utility
- Benefit of this framework: decision criteria are explicit and subject to discussion (with all participants)

Progressing or Stopping?

- Continuous progressing until system goals achieved
- Vague customer purposes: early prototypes, keep options
  - *Firm commitments are best made after the prototype works*
  - *Hang on to the agony of decision as long as possible*
- Untestable performance (ultraquality, hostile environment...)
  - Byzantine failure testing

Conclusion

- Help in organizing the architecting process: progression
- From abstract to domain-specific