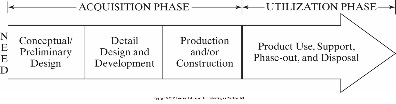
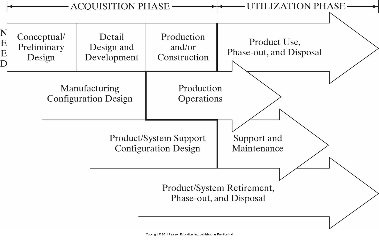
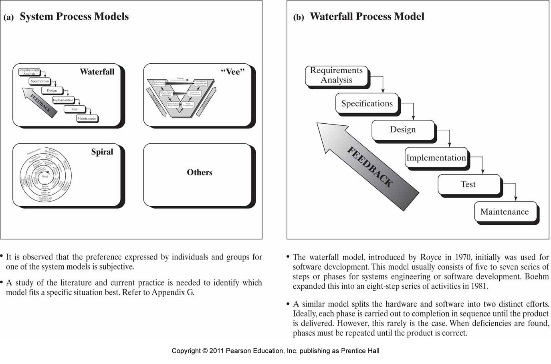
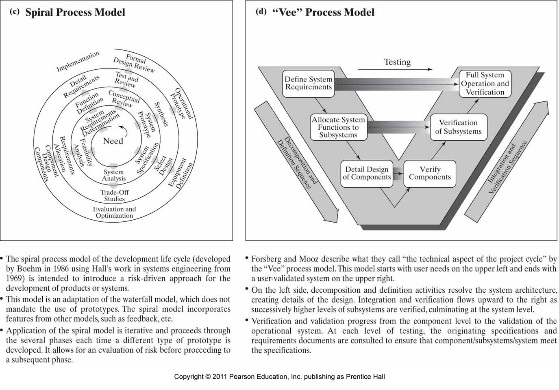
**System**

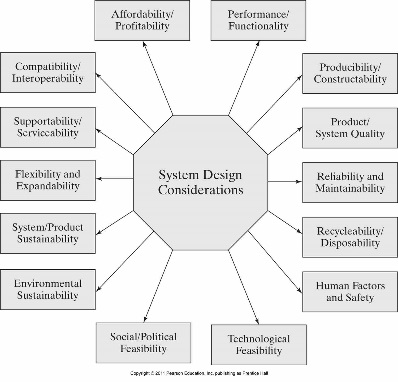
* elements are: a) *components*, b) *attributes*, and c) *relationships*; *alter* either energy, information or materials; can be either *natural*, or *man-*made; can be either *physical* or conceptual; are typically *static* or *dynamic*; set of interrelated components











**Problem statement**

* [relatively] non-technical
* Uses “language” of customer
* Non-complex

**Q’s to get well-made prob. stmt.**

* Q the customer
* Differentiate their needs and wants
* Explore project boundaries
* Do I/O analysis
* Preview user interface
* Survey design attributes
* Identify conflicting needs
* Prepare draft operations manual

**Sections of prob. stmt.**

* Background/overview
* Overall design [not detailed]
* Deliverables

**Prob. stmt. defines**

* Who, what, why (insight)

**Chars. of good prob. stmt.**

* Short, avoids defining soln., not short list of reqs.
* Captures essence of prob./ need, includes qual. & quant.

**Needs analysis**

* When, what (sys. do & primary & secondary functions), where, how often?

**Operational Requirements**

* Mission def, environ. factors, oper. lifetime/deployment, utilization reqs. (freq. of use, hrs. of oper., capacity), perf. params. (mass, volume, range, velocity), effectiveness (MTBM, MDT, MTBF)





**Identifying Technical Measures (TMPs)**

* *Quantitative* values describe performances that stem from op. reqs., are Design Dependent Parameters (DDP) [e.g., MTBF, MTTR, MDT, MTBM, etc.], include weight, envelope, availability to perform/function as designed

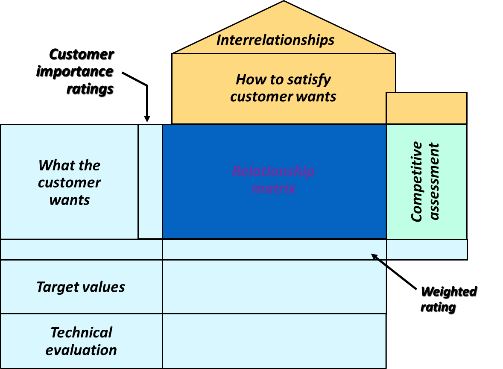
**Quality Function Deployment (QFD)**

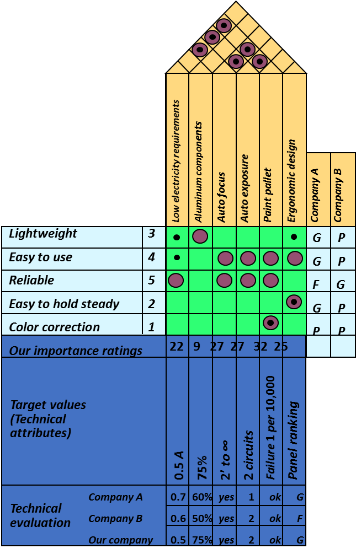
* (Dated) method to identify & capture prime desires of customer
* Uses cross-referenced matrices to perform comparisons of technical & design features resulting in weighted values for TMPs
* *Goal*: identify what customer wants, *predict*: how good/service/process will satisfy customer wants, *relate*: cust. wants to sys. hows, *evaluate*: importance ratings/values, *compare*: performance to desirable TPMs
* *Leads* designers into *preliminary design phase*, *paves* way for functional req. devel. (FFBDs), *allows* for functional allocation by partitioning major sys. elems., *permits* ident. of HW & SW items that multiple functions -> simplify overall sys. arch.

**System Design Flow**

* Abstract notion -> systemic form/function -> repeatable, producible system/product
* 4 design stages: concept.; system; prelim; critical (w/reviews)
* Design revs.: formal method of checking proposed design, provide common baseline for all project stakeholders, permits solving of any interface issues, creates record of design decisions & rationale, leads to greater prob. of a mature design for when production starts
* Reduces risk associated w/prod. phase w.r.t. meeting reqs.

**QFD “House of Quality”**





**Requirements**

* Statement that identifies a capability or function that is needed by a system to satisfy its customer’s needs
* Customer’s needs: solve prob., achieve obj. (cntrct, stanrd, spec)
* 3 sys. eng. Tasks: get reqs; design; verify
* No two sys. are alike but uniform & identifiable process for logically discovering reqs.
* *Process is iterative*: reqs. emerge throughout sys. life cycle and change, customer: “I’ll know it when I see it”, stakeholder priorities change, low-level -> other reqs. become priority

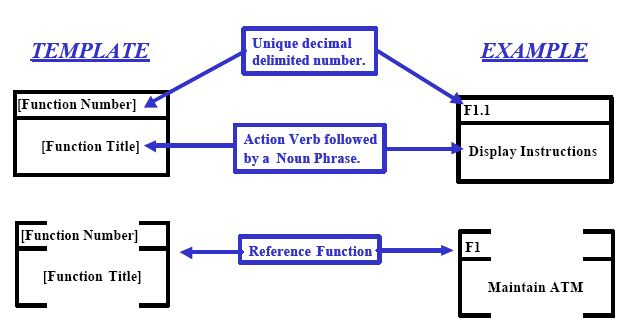


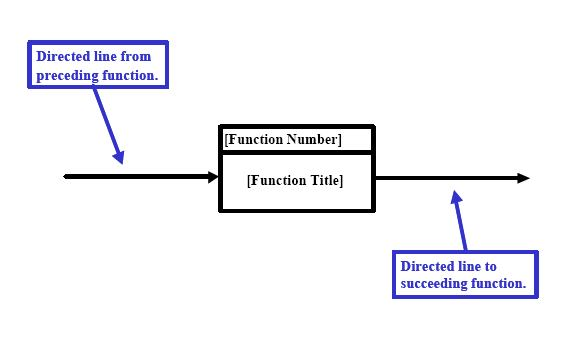
**Functional Requirements**

* What, how well, under what conditions one or more inputs must be converted to one or more outputs @ boundary in question to satisfy customer’s needs. Top-level reqs., basis for reqs. mgmt., the “what” of the system, organic/living, increases in complexity/detail as reqs. decomp. progresses

**Functional Flow Block Diagram (FFBD)**

* Captures functional reqs., time-seq. of operational & support seqs., organized by order of execution, can use Boolean logic
* Benefits: ident. pot. areas for trade study (AOO); dev., analyze, & flow down reqs. to lower levels; provide better pic. of how sys. under dev. is intended to function/work; ident. possible areas for oper. simplification w/in overall system







Typical spec. tree/hierarchy

**Key Specification Terms**

* **Hierarchy**: order of prec. or import. of each spec. in entire compliment of spec. docs. (us. in tree format); **flow down**: act of distributing reqs. from higher prec. docs. to lower level -> greater req. detail; **traceability**: ability to follow interconn. between reqs. w/in partic. spec. & other specs. either higher or lower in prec.; **config. mgmt.**: process of maint. accurate records & current vers. of all of a prog’s specs., drawings, & key docs.

**Standard Specification format**

* **1.0**: intro, purpose, scope of spec.; **2.0**: applicable docs., stds., reference docs; **3.0**: spec. reqs.; **4.0**: verif. of reqs.; **5.0:** packaging/notes, etc.

**Example of System Spec. Req.**

* GPM **shall** enable accountants to generate following financial reports:

Feedback summary; fees summary; invoice summary; listings summary; user summary

* GPM **will** provide following capabilities

**Example of Development Spec.**

* End-User Characteristics, & General Constraints

**Product Specification Example**

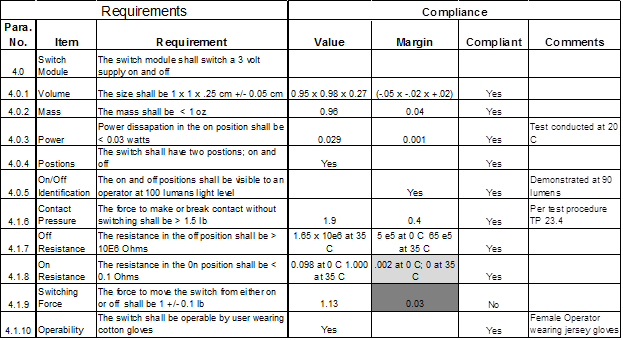
* Physical and Operational Characteristics:

*Performance Reqs*.; *Safety*

**Process Specification Example**

* Product Mgr.; Project Mgr.; Reqs. Eng.; Developer

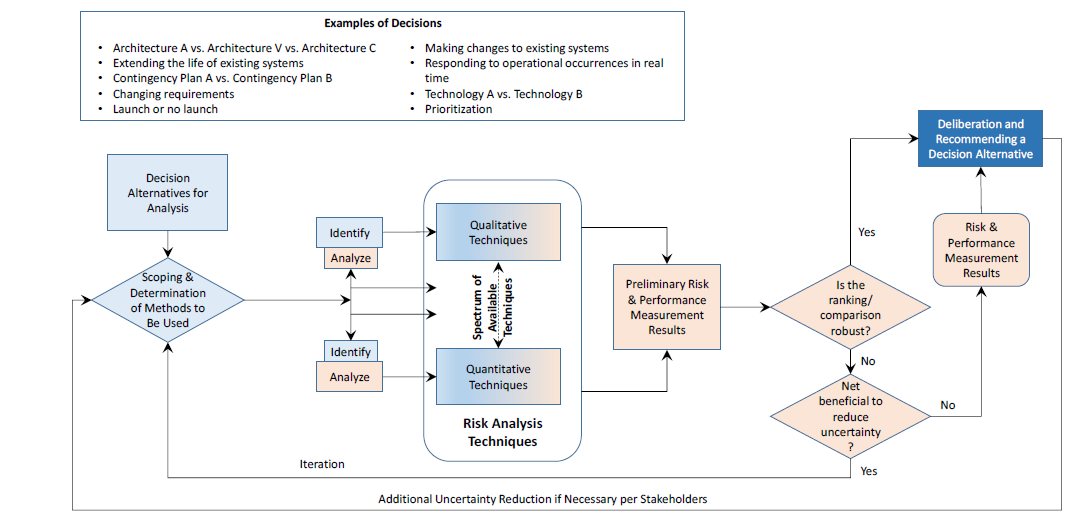
**Sample Verification Cross-Reference Matrix**



**Why Perform a Trade Study?**

* **Gather & assess** possible design sols. for new sys. that one is creating; **determine** whether to make/buy needed asset/tool/SW app.; **methodically eval.** attributes & chars. of each pot. design in light of sys.-level reqs.; **reduce** uncertainty & risk assoc. w/sys. dev.; **arrive** @ decision as to which pot. design sol. will provide “balanced” approach that results in sys. that’s both realizable & functional

**Decision Process Map**



**Examples of How Trade Studies Can be Applied**

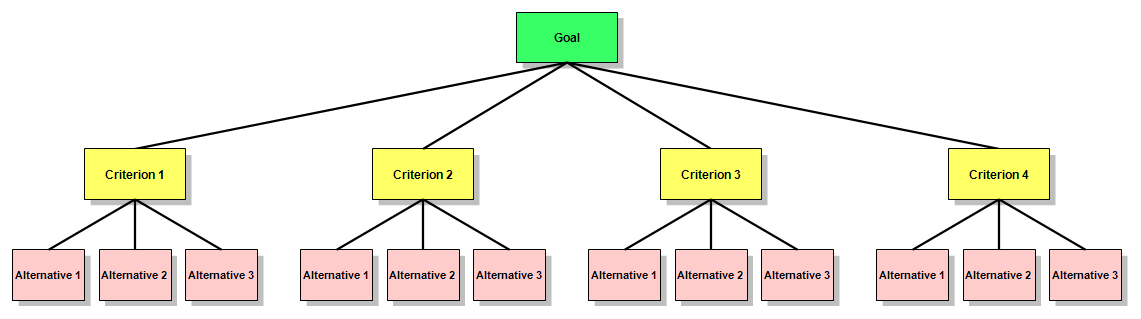
* **Choosing** how/where to allocated limited resources/budget w/in project; **decide** whether to proceed to next checkpoint/gate in sys. dev. process; **what time** to conduct required sys. project revs. (SRR, PDR, etc.); **select** which risk mgmt. methodology to apply to sig. proj. risks to sys. creation while considering costs, schedule, & tech. perf. of the emerging threat

**Means to Conduct a Trade Study**

* *Qualitative*: simplest, easiest; uses ling. descriptors to compare alts./attrs.; downside is possible subjective bias when making comparisons
* *Quantitative*: pairwise comparisons, decision trees, or AHP; advantage is less prone to analyst bias

**Analytic Hierarchy Process (AHP)**

* Pairwise comparisons of attr./chars.; uses matrix/lin. alg. to arrive @ opt. choice when considering (Mult.-Crit./Mult.-Alt.) MCMA descisions
* *Objective*: sits at head of hierarchy; *criteria*: upon which analysis will be performed (each has weight/importance factor as % & add up to 1); *alternatives*: different choices
* Info decomposed into *hierarchy of alts. & dec. crit.* -> *rel.* rank of alts.; quant. & qual. data can be compared; derive weights & priorities



**Handling Costs – Diff. From Economic Evaluation Method(s)**

* Dangers of including cost as another criterion: political & emotional response? *Separate* benefits and costs into different hierarchical trees.
* Cost/Benefit eval.: choosing alt. w/best bens./cost ratio. Use RoR/BEP to properly consider decision from a cost POV

**System Verification**

* *Validation*: process of confirming that a set of reqs., design, or sys. *meets the intent* of dev. or cust. (does the *sys*. meet its req.?). **Tends** to be oriented toward analysis. **Takes place**: before design & during detailed design (ideally *before* Sys. Reqs. Rev.). **Important** b/c when verifying, want to verify the *right* reqs. & changing reqs. late neg. impacts cost & schedule. Determines correctness/completeness of end product (ensures actual needs of stakeholders are met)
* *Verification*: process of confirming that a req. or sys. is *compliant* (does the sys. meet *this req.?*). Ensures sys. complies w/sys. reqs. & conforms to its design. Done in Phase D (integration, right leg of Vee diagram). Prelim. Plan for how reqs. will be verified is created when the reqs. are gen by CDR -> prelim. verif. matrix. Reviews precede each key decision point. About confidence.
* *Verification matrix* specifies: req. statement; any assoc. verif. req. including “successful criteria”; method of verif. (inspection, analysis, demo, test); results of verif. as they become available.
* *Verification plan*: documents a project’s approach to verification (people, schedule, equip., facilities). Includes sys. qualification, site verification, post-op/disposal verification; needed support equipment
* *Methods*: inspection, analysis, demo, test (most important)
* *Inspection*: determine conform. to reqs. by visual exam. of drawings, data, or item using std. qual. ctrl. methods (no spec. lab procs./equip.)
* *Analysis*: eval. of data by gen. accepted analytic techs. to determine that item will meet reqs. (sys. eng. analysis, statistics, qual. analysis, similarity, sims.). Used when test or demos can’t adequately or cost-effectively address all conditions the sys. must perform can’t be shown to meet reqs. w/o analysis
* *Demo*: determines conform. to reqs. by operation, adjustment, or reconfig. of test article. Generally verify sys. chars such as human eng. features, services, access feats., transportability. Relies on observing & recording functional op. *not* requiring use of elab. instrum., spec. test equip., quant. eval of data
* *Test*: uses tech. means (spec. equip, instrum., sim. techs., etc.). Operation of all/part of sys. under limited set of controlled conditions to determine that *quantitative* design/perf. reqs are met. Analysis of data from tests shouldn’t be confused with the *analysis* method. Preferred when: analytic techniques don’t produce adequate res., failure modes exist, any components directly assoc. w/critical sys. interfaces. Space sys. go through rigorous ground-tests to sim. Launch/space evirons.
* Using heritage designs can save development time and money, but they should be validated and verified as if they represented new hardware. Simple procedural errors - like borrowing another project’s bolts - can lead to multi-million-dollar (or life threatening) accidents

**Risk Management (RM)**

* *Risk*: uncertain event or condition that, if it occurs, has a + or - effect on at least one project objective, such as time, cost, scope or quality
* *Primary risk management processes*: **identification** (stages: candidate, in process, baselined/active, closed/mitigated); **assessment/analysis** (how significant is this risk? Likelihood & impact. Prob. Impact Diag.); **planning/handling** (accept, transfer, mitigate, avoid); **control** (tracking); **communication**. *Fundamental methods of assessment:* qual/quant
* *Qual. risk assessment*: most prevalent form of RM (simple, intuitive, leverages SME knowledge from exp. IPT members). *2 Params*: risk consequence; likelihood of occurrence L­­O). *Formula*: consq x LO
* *Quant*. *RM (PRA)*: not use that much. Not meant to manage/determine status of prog. risks. Tool of system of safety and reliability eng. disciplines (predictive tool vs. descry.). Addresses random/probability. Requires sig. knowledge of stats/prob. theory.
* *Tools:* Monte Carlo Method (MCM) predicts range of outcomes/scenarios using rand. num. algo. to produce Gaussian dist. func., indicating probability of events outcomes
* *Risk vs. Issue*: **risk** in future, **issue** in past
* *Acceptance*: accept risk & its consequences part of doing business, *transfer*: move risk item to either another discipline or owner/team (doesn’t mean refusing risk), mitigate (apply program resources to lessen risk and/or impact), avoid (change req., alter design, something for risk to go away)

**Systems Engineering Work Model**

* Evaluation, synthesis, analysis, go back to 1

**Extra Vocabulary**

* *Components*: parts of a system; *attributes*: characteristics of a system; *relationships*: how system components interact

**House of Quality (HOQ):** way to pull out customers' needs and find out what they want

**Technical Performance Measures (TPMs)**: *quantitative* values that describe system performance

* stem from system operational requirements
* Design Dependent Parameters (DDP): when design is finally converged on, parameters drive what performance measurements (e.g., failure weight)
* can include: weight, envelope, and availability to perform/function as designed

**Quality Function Deployment (QFD)**

* somewhat dated (although still used)
* weighted values for TMPs using cross-referenced matrices
* primary goal: identify what customer wants
* how to get what customer wants (good/service/process)
* relates wants to ... hows?
* takes place in preliminary design phase of system development

**System Design Flow**

stages (each includes an associated review) (conceptual design, system design, preliminary design, critical design)

**Three Systems Engineering Tasks**: get reqs. design, verify reqs.

**System Requirements**

* Uniform & identifiable process for discovering system requirements
* Requirements emerge throughout system life-cycle (i.e., they can change)

**Requirements Change**

* Can't get all requirements upfront
* Requirements are emergent (2nd bullet point above)
* Customer, "I'll know when I see it"
* Customer/stakeholder priorities change over time
* Low-level requirements -> ...

**Customer Usually Misstates Needs**: May try to arrive at solution before thinking about problem

**Types of Requirements**

*Primary*: comes directly from a contractual agreement/commitment from the customer. *Derived*: comes internally from project/organization that's developing the system

**Requirement Application**

**How is a particular requirement applied to the system that is being developed?**

* *Product*: imposed on the "physical" product. *Qualitative*: description of what system is going to do. *Quantitative*: e.g., mean time to failure
* *Program*: applied on the program (concerned with execution). *Tasks*: developers must do "something." *Compliance*: how are we going to verify that a requirement has been met (objectively): *Mandatory* -> "Shall" (must have it, otherwise in violation of contract); *Guidance* -> "May" or "Will" (nice to have); *Information* -> "Should"; *Regulatory*: meeting laws/rules

***Dog House Example***

Your company is developing a dog house. Come up with some requirements.

Qualitative, *Quantitative*: The dog house shall remain standing during a Category Two storm on the Saffir-Simpson scale; the dog house shall remain standing after a 6.0 earthquake on the Richter scale

**Requirement Quality:** Complete, lack of ambiguity; scope (avoid motherhood statements “dog shall enjoy dog house”)

**Types of Risk:**

* Known known (tip of the iceberg): known risk, probability of occurrence known
* Known unknown (just beneath the water line): worst. Someone in org. knows that some risk exists but doesn’t say anything
* Unknown known (1/3 of the way under): unknown risk, prob. unknwon
* Unknown unknown (bottom 1/3)