Software Technology Support Center (STSC)

Helping Government Organizations Buy and Build Software Better

What is this System Really Suppose to do?

Or the “You Must be Kidding Syndrome.”
LESLIE (LES) DUPAIX
USAF STSC
les.dupaix@hill.af.mil
System analysis steps

1. Identify the customers need
2. Evaluate the proposed system for feasibility
3. Perform economic and technical analysis
4. Allocate function to hardware, software, people and data (database)
5. Establish cost and schedule constraints
6. Create a system definition
1. Identification of need

- Meet with customer and end-user
- Understand who the “real” customers are (and uncover political agendas)
- Understand the requirements
  - Know the difference between customer requirements and desirements
- Establish goals
  - Is current technology adequate to meet goals
  - What is potential market
  - How does system integrate with existing constraints
- Document in a *System Concept Document*
2. Feasibility Study

- Related to Risk Analysis and Risk management
  - Development risks
  - Resources availability
  - Technology availability
  - Security

- Evaluate feasibility of
  - Economic feasibility
  - Technical feasibility
  - Legal feasibility
  - Examine other alternatives
3. Economic Analysis

- Perform Cost/Benefit analysis
- Lots of factors to look at
  - cost reduction (CR)
  - error reduction (ER)
  - increased flexibility or capability (IF)
  - increased speed (IS)
  - improvement in management planning and control (MC)
  - security
3. Technical analysis

- Assessment of the technical viability of the system
  - Is technology available?
  - What new materials, methods or processes are required?

- Tool available
  - Models and simulations
  - Probability theory
  - Queuing theory
  - Control theory
How to do technical analysis

- **Build a System Model**
  - Simple enough to understand, but as close to reality as possible (to yield valid results)
  - Highlight factors that are relevant or important, and (with discretion) suppress those not as important
  - Include relevant factors, and should give repeatable results
  - Small enough to be timely. If too big, consider breaking down into many smaller models
  - Make the model expandable and modifiable. This allows “tuning” of the model, and also expansion and inclusion of changing requirements
4. Allocate functions to...

- Hardware
- Software
- People
- Data (database)
- Other system elements
- Security

- Basically, an architectural model
5. Establish cost and schedule

Based on
- customer needs
- economic feasibility
- technical analysis
- functional allocation
- Security

Requires both management and customer buy-in

Critical factor is typically time, not cost
6. Create systems definition

- The “blueprint” that guides the entire systems development.
- Explains what each area (hardware, software, etc) is responsible for.
- Explains interfaces between areas.
- Forms the *Systems Specification*, which is the basis for future
  - hardware engineering
  - software engineering
  - database engineering
  - human engineering
  - Security engineering
Modeling the system

- It is necessary to define the boundaries of the system.
- Typically, a graphical representation or model is best for “first cut”
- Create different models or views of the system (operational view, system view, and technical view)
- DODAF is way to illustrate architecture
- An ACD (architecture context diagram) is another a high-level diagram that shows boundaries between the system and its’ environment. It lists external interfaces (informational boundaries)
Linkages Between the Views

Operational View
- Identifies Warfighter Relationships and Information Needs

Technical View
- Prescribes Standards and Conventions

Specific Capabilities Identified to Satisfy Information-Exchange Levels and Other Operational Requirements
- Technical Criteria Governing Interoperable Implementation/Procurement of the Selected System Capabilities

Systems View
- Relates Capabilities and Characteristics to Operational Requirements

1: Systems Interface Description

1: Technical Standards Profile

1: High-Level Operational Concept Graphic

From AFIT SYS 283
Introduction to Architecture
The User Acceptance Test Plan should be used to record the users' story of the systems' use and the user's expected output.
## Software Engineering

<table>
<thead>
<tr>
<th>Reqs. Analysis</th>
<th>Design</th>
<th>Implement</th>
<th>Test</th>
<th>Maintain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan for Maintenance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configuration Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verification and Validation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Quality Assurance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Risk Analysis &amp; Risk Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Engineering Project Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Improvement</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Lifecycles

### Strengths and Weaknesses

<table>
<thead>
<tr>
<th>Capability</th>
<th>Pure Waterfall</th>
<th>Code-and-Fix</th>
<th>Spiral</th>
<th>Modified Waterfall</th>
<th>Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poorly understood requirements</td>
<td>Poor</td>
<td>Poor</td>
<td>Excellent</td>
<td>Fair to Excellent</td>
<td>Excellent</td>
</tr>
<tr>
<td>Poor Architecture</td>
<td>Poor</td>
<td>Poor</td>
<td>Excellent</td>
<td>Fair to Excellent</td>
<td>Poor to Fair</td>
</tr>
<tr>
<td>Highly Reliable System</td>
<td>Excellent</td>
<td>Poor</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
</tr>
<tr>
<td>System Growth Built in</td>
<td>Excellent</td>
<td>Poor to Fair</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
</tr>
<tr>
<td>Risk Management</td>
<td>Poor</td>
<td>Poor</td>
<td>Excellent</td>
<td>Fair</td>
<td>Fair</td>
</tr>
<tr>
<td>Predefined Schedule</td>
<td>Fair</td>
<td>Poor</td>
<td>Fair</td>
<td>Fair</td>
<td>Poor</td>
</tr>
<tr>
<td>Midcourse Correction</td>
<td>Poor</td>
<td>unknown</td>
<td>Fair</td>
<td>Fair</td>
<td>Excellent</td>
</tr>
<tr>
<td>Customer Visibility</td>
<td>Poor</td>
<td>Fair</td>
<td>Excellent</td>
<td>Fair</td>
<td>Excellent</td>
</tr>
<tr>
<td>Management Visibility</td>
<td>Fair</td>
<td>Poor</td>
<td>Excellent</td>
<td>Fair to Excellent</td>
<td>Fair</td>
</tr>
<tr>
<td>Low Management and developer skill level</td>
<td>Fair</td>
<td>Excellent</td>
<td>Poor</td>
<td>Poor to Fair</td>
<td>Poor</td>
</tr>
<tr>
<td>Low Overhead</td>
<td>Poor</td>
<td>Excellent</td>
<td>Fair</td>
<td>Excellent</td>
<td>Fair</td>
</tr>
</tbody>
</table>

Rapid Development (McConnell, 96)
## Change Possibility

<table>
<thead>
<tr>
<th>Phase</th>
<th>Effort/Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFI</td>
<td>4x</td>
</tr>
<tr>
<td>RFP</td>
<td>2x</td>
</tr>
<tr>
<td>Contract</td>
<td>1.5x</td>
</tr>
<tr>
<td>Award</td>
<td>1.25x</td>
</tr>
<tr>
<td>Req. Review</td>
<td>1.0x</td>
</tr>
<tr>
<td>Design</td>
<td>0.8x</td>
</tr>
<tr>
<td>Code Review</td>
<td>0.67x</td>
</tr>
<tr>
<td>Test</td>
<td>0.5x</td>
</tr>
<tr>
<td>Delivery</td>
<td>0.25x</td>
</tr>
</tbody>
</table>

**Cost Models for Future Life Cycle Processes:** COCOMO 2.0 (Boehm, 1995)

*Software Technology Support Center (STSC)*
Finding Requirement Errors

Requirements Review 1 hr
Design 2.5 hrs
Code 13 hrs
Test 17 hrs
Deployment 33 hrs
You can control change to only two sides of a triangle; The third side must freely adapt – or else it’s not a triangle anymore.
Software is both a source of amusement and engineering achievement.
The Temple of Software Engineering

Quality

“Pillars” of Good Software

If we all work together, look at what we can build!

GOOD PROCESS

Faster  Cheaper  Better

Software Technology Support Center (STSC)
But just a few stubborn people can ruin the entire process!
More Information

You can go on the internet and search for more information on the topics discussed and/or check on the links below:

www.dau.mil

www.nps.edu

www.afit.edu/about.cfm

http://www.usability.gov/templates/docs/

u-test_plan_template.doc

www.uml.org