Automation of Provisioning

Larry Bernstein
Have Laptop-Will Travel
lbernstein@ieee.org
Phone: 973-258-9213

Provisioning Functions

1. Accept, edit and process service orders.
2. Interpret service order -- translate it to a request for "so many pairs at such-and-such an address."
3. Assign the pairs -- find unused pairs for the service; rearrange existing telephone company pairs as needed.
4. Assign central office equipment -- connect the assigned pairs to the main distributing frame.
5. Request telephone company people to make the needed connections if none already exist.
6. Activate dial tone.
7. Rearrange telephone company plant as new pairs are installed in anticipation of neighborhood growth.
   - Assign new pairs when repair is needed

Goal: Flow-Through Provisioning

**Benefits of Automation**

- Technicians
- System Administrator
- Developer

System
- Hardware
- Software
- Training
- Procedures
- Tools

Cost of System

\[ \text{Technicians} - \text{System Administrator} + \text{Developer} = \$100 \text{m/yr} \]
Staffing

<table>
<thead>
<tr>
<th>Department</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applied Research</td>
<td>10</td>
</tr>
<tr>
<td>System Engineering</td>
<td>50</td>
</tr>
<tr>
<td>Development</td>
<td>200</td>
</tr>
<tr>
<td>Support</td>
<td>15</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>275 people for 5 years</strong></td>
</tr>
</tbody>
</table>

Project Information

- System broken into 5 subsystems
- Common Project Management
- Independent Test Group
- Independent System Administration Tools
  Group
- Spiral Model used for 2 of the 5 subsystems
Organization Structure

Kruchten’s “4 + 1” Model for Developing Software Architecture

This is an innovative and comprehensive integration of the abstractions needed for writing system requirements.
Project A: *Order Reading and Analysis Software*

Size of Prototyping Effort:
12K lines of C Code (10% of final system module)

Purpose:
Find a method for order reading and analysis, applicable to variable formats.

Duration and Staff of Prototype:

Four people for eight months.

Experience:
1. Final requirements based on prototype results.
2. Alerted developers' to the possibility of a having a tunable system.
3. Early evaluation of functional decomposition and performance showed bottlenecks in the dispatcher.
4. Eliminated the possibility of reusing code from another project.
5. Prototype was thrown away due to decomposition and performance problems.
Project B: Outside Plant Data Base System

Size of Effort: 5% of 500K line of source code.

Purpose:

To evaluate database structures for an outside plant data and to experiment with approaches to handling multiple future states of equipment usage.

Duration:

Three people for 15 months.

Experience:

• A data base structure using hyper graph theory was invented.
• An algorithm for handling both time-driven and event-driven assignments was invented.
• The prototype became the basis for the production code.
• UNIX flat files were used to model loop plant by way of a directed graph.
• The prototype showed database portability from Unix to the Mainframe.
Why bad things happen to good systems

- Customer buys working software product.
- System works with 40-60% flow-through
- Developers make address changes

BUT

- Customer refuses critical Module
- Customer demands 33 enhancements and allows data corruption
- Developers misgauge extent of changes