



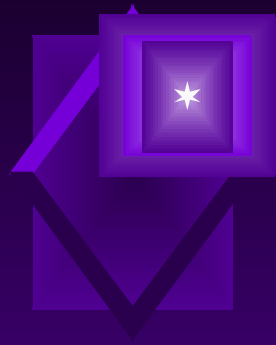
Faster, Better, Cheaper

Lessons from the Commercial World

Steve Chamberlain

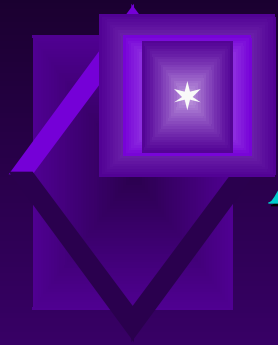
Integral Systems, Inc.

src@integ.com



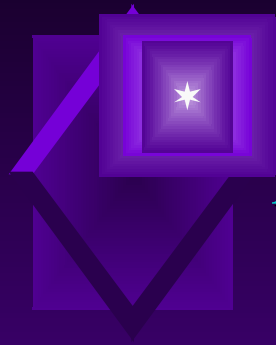
Topics

- ◆ A Successful Commercial Architecture
 - ◆ Decentralization
 - ◆ Decoupling
 - ◆ Dynamic Reconfiguration
 - ◆ Sample Architecture Diagram
- ◆ Axioms, Dogma, And Sweeping Generalizations
- ◆ The Central Dogma
- ◆ How To Build It Faster, Better, Cheaper
- ◆ How *Not* To: Lessons Learned



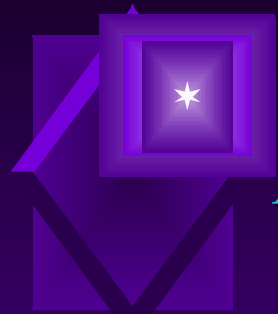
A Successful Commercial Architecture

- ◆ We want to discuss an approach which supports a variety of satellite missions
 - ◆ Currently flying Government and commercial LEO/GEO birds
 - ◆ We'll call it the Hypothetical Control, Operations, and Processing Environment (HCOPE) to keep focused on technical issues
 - ◆ Avoid looking like shameless marketeers
 - ◆ HCOPE is available (under a slightly different name) from a firm in suburban Maryland
- ◆ Key features of the HCOPE architecture (“3-D”):
 - ◆ Decentralization
 - ◆ Decoupling
 - ◆ Dynamic Reconfiguration



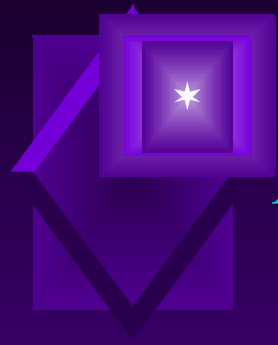
Decentralization

- ◆ All computers are interconnected via LAN/WAN.
All software functions can run on any computer.
 - ◆ No host computer to get bogged down when you add seats
 - ◆ Fine-tune the processing and network load by reallocation
 - ◆ Don't get obsessed with details which are architectural only if the design is poor:
 - ◆ How do I get redundancy and hot backups?
 - ◆ By running another copy of the software
 - ◆ Do I run the archives at the station or the control center?
 - ◆ Both, of course, and as many instances at each as you want
 - ◆ What if the commanding computer goes down?
 - ◆ Have a backup copy running somewhere else take over



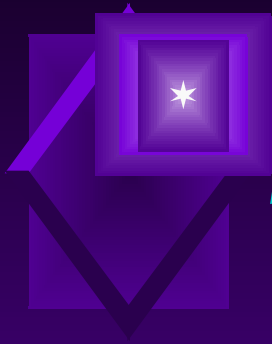
Decoupling

- ◆ Support open systems standards so you're not tied to specific computers or peripherals
 - ◆ HCOPE supports any UNIX platform
 - ◆ Windows NT version coming soon to a control center near you
- ◆ Isolate hardware dependencies with a common layer for interprocess communications
 - ◆ Avoid inadvertently incorporating specific frame syncs, telemetry boxes, etc. in your architecture
- ◆ Define the satellite and ground characteristics in a database
 - ◆ Make changes and reconfigure without rewriting software

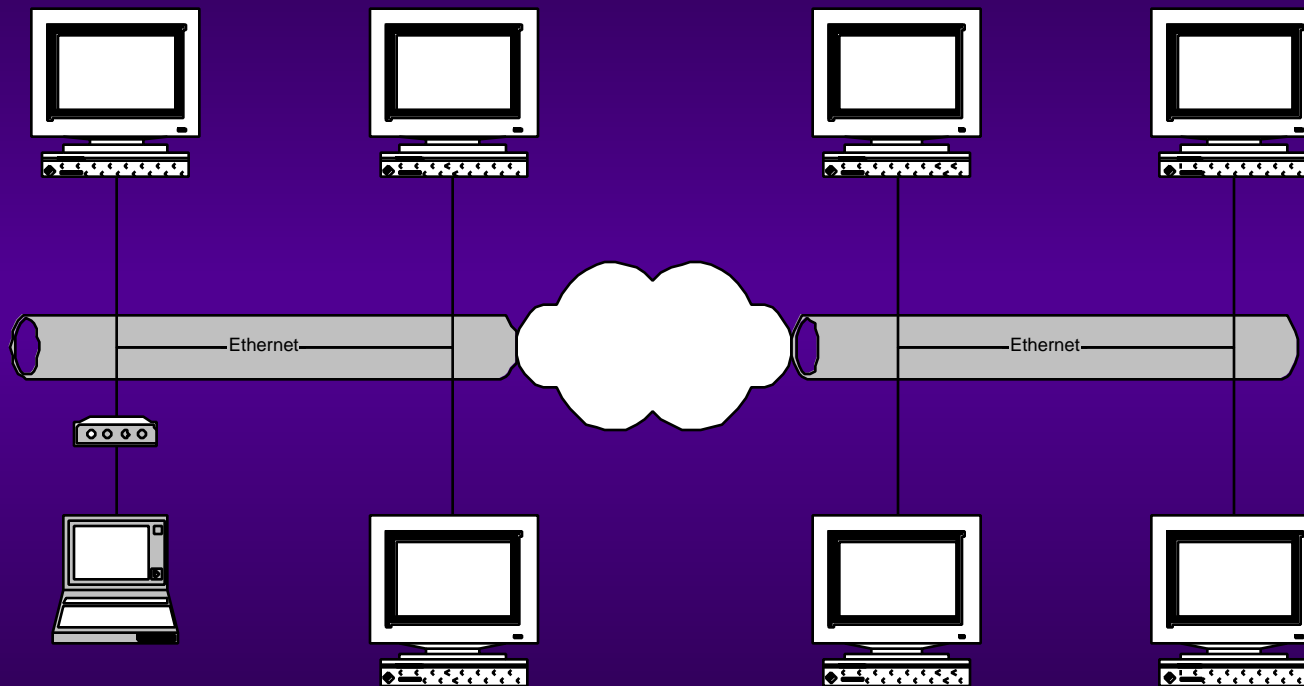


Dynamic Reconfiguration

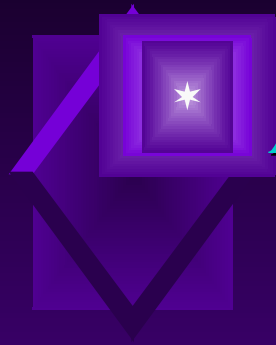
- ◆ Provide an open data service for two-way communications
 - ◆ Let external programs share data, commands, telemetry, etc. with core functions
 - ◆ Allows system to be reconfigured for mission-unique requirements without impacting the core architecture
- ◆ Provide a common event mechanism with external triggers
 - ◆ Allows core software to launch scripts and external programs in case of an alarm



Sample Architecture Diagram

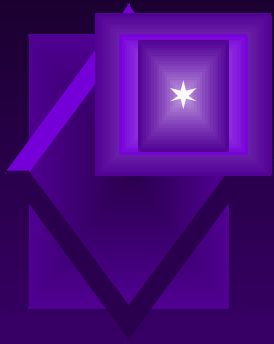


Which computer runs which function? The design should let you reallocate at will!



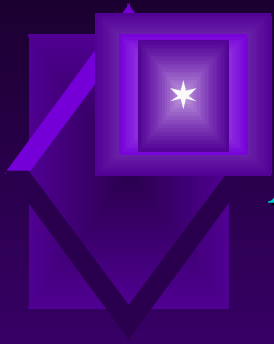
Axioms, Dogma, and Sweeping Generalizations

- ◆ The quality of a control center is inversely proportional to its cost
- ◆ The quality of a control center is inversely proportional to the number of people who show up at the design reviews
 - ◆ AKA “Too many cooks spoil the broth”
- ◆ Building control centers should be easy
 - ◆ Data rates are low, computers are fast, and the functionality is fairly standard



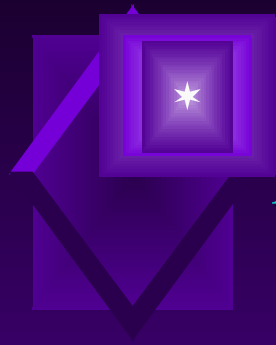
The Central Dogma

\$	Project	Satisfaction
10^9	DSM	0
10^8	DMSP	4
10^7	PACS	7
10^6	HCOPE	9



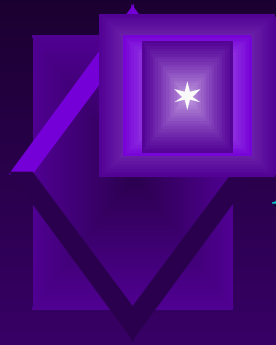
How To Build It Faster, Better, Cheaper

- ◆ Adopt a set of core functionality which is the same for all missions
 - ◆ Handle mission-unique requirements with external programs which are not coupled to your core architecture
- ◆ Award development contracts competitively on an FFP basis
 - ◆ A market economy beats a planned economy every time
 - ◆ You'll get better systems and better architectures for a cheaper price through competition alone, without a lot of bureaucratic process controls



How To Build It (Continued)

- ◆ Keep attendance at design reviews and working groups to a minimum
 - ◆ Let the marching army review the material and submit comments in writing
 - ◆ The effort involved in writing will cause self-filtering of suggestions
 - ◆ Otherwise, you'll end up throwing in the kitchen sink
- ◆ Above all else, **KISS!**
 - ◆ Avoid requirements creep like the plague itself



How Not To: Lessons Learned

- ◆ Don't dictate an architectural standard
 - ◆ Else industry will pass you by: NASCOM, TPOCC
- ◆ Don't dictate a tools standard
 - ◆ Else industry will pass you by: ADA, FORTRAN
- ◆ Don't let blanket support and cost plus contracts
 - ◆ Else you eliminate competition and run afoul of the central dogma: DSM, TPOCC
- ◆ Don't have a marching army at your reviews
 - ◆ Else KISS is doomed