Architecture skeletons: Lessons-learned from large-scale software-intensive system development

10 March 2010

Neil Siegel
Sector Vice-President & Chief Engineer
Member, National Academy of Engineering

NORTHROP GRUMMAN

Agenda

- What kind of systems are we talking about?
- What are the typical symptoms of serious development problems for these types of systems?
- What are the underlying root causes?
- One approach to a solution
- Findings

Example system:

Battlefield digitization (aka "Blue Force Tracking")

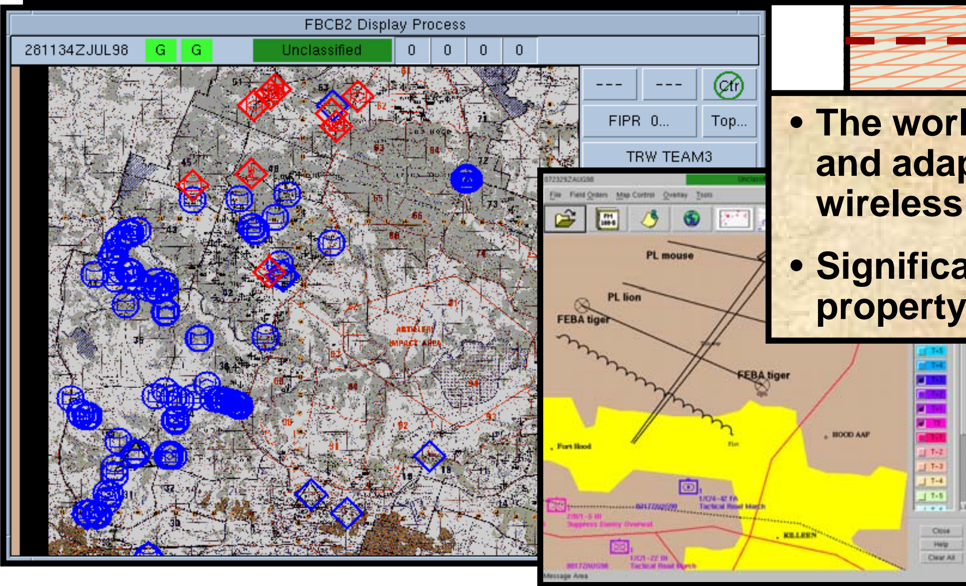
NORTHROP GRUMMAN

The use of ***information technology*** to increase the effectiveness of the U.S. land combat team

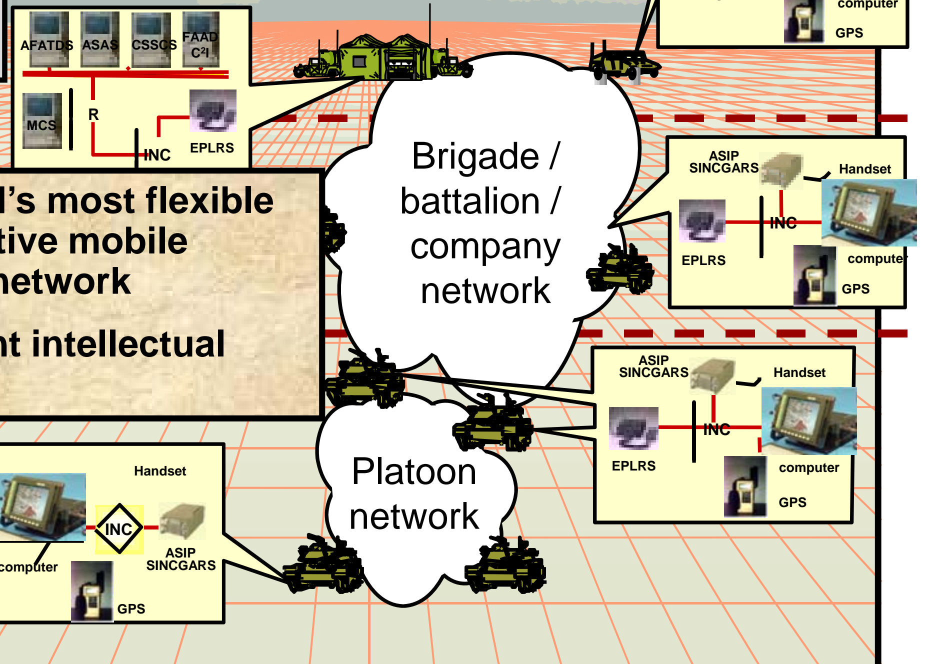
- **Where am I?**
- **Where are my buddies? My allies?**
- **Where is the enemy?**
- **Where are the dangerous areas?**
- **What is the route over which I am to move?**

On the move . . . all the time

Every Army platform . . . wheels, tracks, rotary wing, dismount



- **The world's most flexible and adaptive mobile wireless network**
- **Significant intellectual property**



Other examples

- U.S. Forward-Area Air Defense System
- Counter-RAM (rockets, artillery, mortar) system
- Tactical High-Energy Laser
- Command Center Processing and Display Systems
- Various large radar systems
- Logistics automation systems
- Air traffic control systems
- Many others

What are the typical symptoms of development problems for these types of systems?

GRUMMAN

- Performance &/or capacity is substantially below that required
- Response-time is substantially worse than that required
- Availability &/or mean-time-between-failure is substantially below that required
- External interfaces do not work as specified

The key lessons-learned about the root causes underlying the above:

It's the "unplanned dynamic behavior".

If you don't design with these risks in mind, you might not see the problem until very late in the program.

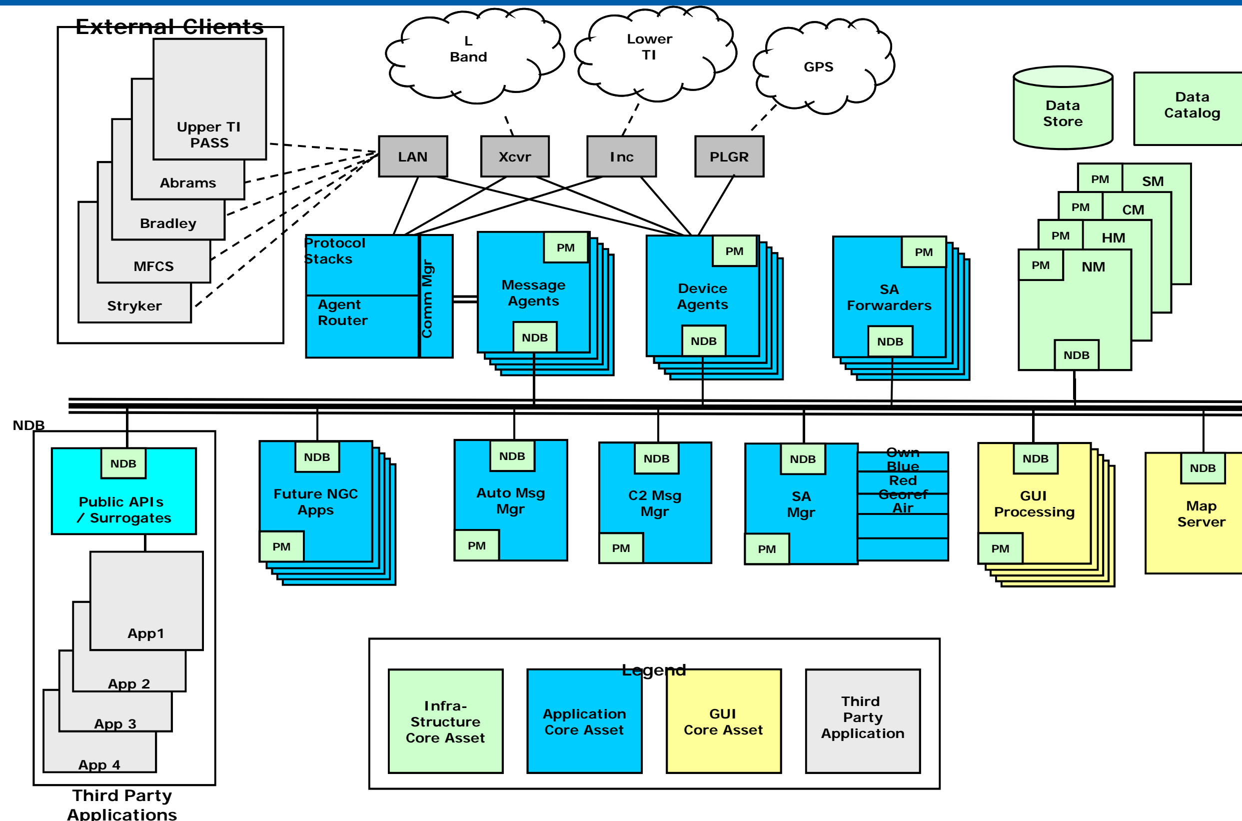
NORTHROP GRUMMAN

Programmatic needs

- Need to start analyzing and debugging dynamic behavior much earlier in the life-cycle
- Enable implementation and exercising of actual external interfaces much earlier in the life-cycle
- Facilitate prototyping and benchmarking, thereby providing more visibility into progress
- Provide technical tools that allow us to meet software schedules and costs
- Put something into the design that allows us to consider the performance, capacity, response-time, and availability goals
- And in general, lower the risk that bad things don't show up until systems integration

NORTHROP GRUMMAN

One approach: the system architecture skeleton (SAS)



- Build and exercise the system architecture skeleton:
  - Defines and implements all processing paths through the system.
  - Mechanize the threads through some reasonably deterministic tool, allowing one to prevent unplanned dynamic behavior.
- Populate the SAS at first with stubs, models, and prototypes
- Integration consists largely of replacing these with actual products

NORTHROP GRUMMAN

Findings

- Seems to help
  - Many successes, over many years, in many different customer contexts
- Not strongly dependent on any particular implementation technology
  - Core ideas have been re-implemented in several successive generations of technology
- Appears to decrease integration risk by:
  - Increasing the time available for performing integration
  - Allows the implementation, integration, and debugging of the system control flows largely separate from the functionality

Questions?