Software Architecture in Practice

Chapter 11: Air Traffic Control - A Case Study in Architecture for Availability

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Sponsored by the U.S. Department of Defense

Lecture Objective

This lecture will enable students to understand the architectural techniques used to achieve a high-availability system.

Air Traffic Control System

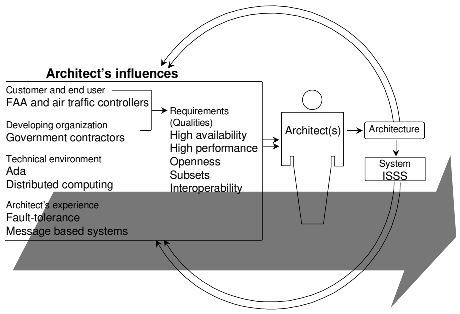
The Initial Sector Suite System (ISSS) was intended to replace the old display consoles and software in the 22 en route centers in the continental United States.

Its customer was the Federal Aviation Administration (FAA).

It was the first phase in a planned series of replacements.

- **tower systems**
- **ground control systems**
- **transoceanic systems**

ABC for Air Traffic Control System



System Requirements - 1

Interface to FAA's HOST computer system
- **central database of all flights and flight plans**
- **source of processed radar data**

Display radar, flight plan data on consoles

Be configurable into sector suites
- **one to four stations per sector**
- **two consoles/controllers at each station**
  - **flight plan**
  - **radar data**
- **completely reconfigurable**
- **monitor and control consoles also**

System Requirements - 2

Up to 210 consoles per installation

400-2440 separate aircraft tracks

16-40 radars

Must provide recording/playback capability

Must handle conflict alerts

Initial Quality Requirements -1

Ultrahigh availability
- 99.999% available: down fewer than 5 minutes per year
- failures with 10-second recovery time don't count against total

Hard real-time performance: radar data

Architectural compatibility with future replacement systems (e.g., tower, ground)

Initial Quality Requirements -2

Usability: human/computer interface critical

"Openness:" ability to import COTS software

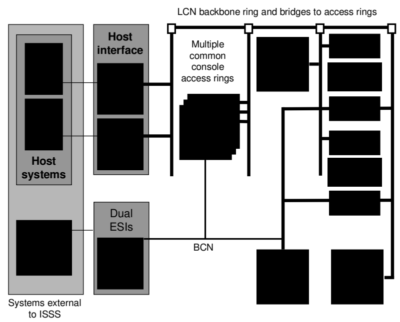
Portability: use new computers, networks

Subsequent Quality Requirements

Ability to field a subset: When cost/schedule overruns forced a cutback, a functional subset was needed.

Interoperability: As new FAA programs come online, integration with ISSS system is important (collision avoidance, free flight).

ISSS System Architecture



ISSS Software Components

Computer software configuration items (CSCIs)
- **mandated by U.S. government standards**
- **basis for teams: work assignments/modules**

Five CSCIs
- **display management: 350 KSLOC**
- **common system services: 345 KSLOC**
- **recording, analysis, and playback: 31 KSLOC**
- **National Airspace System modification**
- **IBM AIX operating system**

Software onto Parallel Hardware - 1

Processors were combined into *processor groups* (e.g., supporting a sector suite).

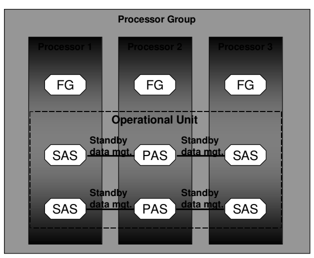
Each processor group hosts separate copies of one or more *applications*.
- **A fault-tolerant application is an *operational unit*. One executing copy is *primary*, the others are *secondary*.**
- **A non-fault-tolerant application is a *functional group*. Copies run independently on separate processors.**

Software onto Parallel Hardware -2

Applications interact with each other in client-server fashion.
- **client sends server *service request message***
- **server responds**
- **application may be client or server or both**

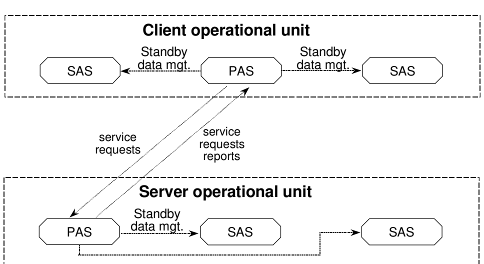
Within operational unit
- **primary sends periodic data update message to secondaries**
- **secondaries notice if primary fails and one takes over**
- **new primary sends message to clients establishing status (e.g., "Were you waiting for something?")**

Fault-Tolerant Software Scheme



FG = functional group
PAS = primary address space (primary part of operational unit)
SAS = secondary address space (backup part of operational unit)

Client-Server Applications



Template for Fault Tolerance -1

loop
- *get event*
- *case event_type*
  - *normal request for processing? Carry out request and send state data update to other address spaces*
  - *terminate? Clean up and stop.*
  - *state data update? We must be Secondary. Accept and update.*
  - *switchover? check with clients…*
  - *others? log error.*
- *end case*

end loop

Template for Fault Tolerance -2

The same code works for PAS and SAS.

A system-wide fault tolerance scheme accrues the same benefits as encapsulation, without encapsulation.

Whole classes of programming errors are eliminated.

Other fault tolerance mechanisms
- **system-wide error reporting/handling policy**
- **handled at lowest (earliest) level possible**
- **warm restart (uses checkpoint data)**
- **cold restart (uses default data)**

Assessing the Architecture

ISSS was drastically over budget and behind schedule.

FAA wanted to know if anything was salvageable; the review team assessed the system for the quality attributes listed earlier.

SAAM Scenarios

Scenarios

- **import third-party air traffic control software**
- **use X Windows for GUI**
- **upgrade to a new, faster network**
- **double flight capacity requirement**
- **change a flight controller's display format**
- **add a new operational unit**
- **delete requirement to support flight strips**

Result of SAAM: ISSS met its objectives to a high degree

ISSS Outcome

ISSS was eventually dropped in favor of a smaller, cheaper system that used the same architecture, but which did not have the strict fault tolerance requirement.

The architecture was clearly the key to ISSS: Multiple styles were evident.

- **layered**
- **cooperating process**
- **client-server**
- **message-passing**
- **object-based**

Lecture Summary

ISSS achieved its quality goals in a variety of ways.

Goal	How achieved
High availability	hardware redundancy (both processor and network) software redundancy (layered fault detection and recovery)
High performance	distributed multiprocessors, front-end schedulability analysis, network modeling
Openness	interface wrapping, layering
Modifiability	templates, table-driven adaptation data
Ability to field subsets	appropriate separation of concerns
Interoperability	client-server division of functionality message-based communications

Discussion Questions

1. High reliability was a main impetus behind the architecture presented in this lecture. How were other quality attributes, such as performance, affected by this requirement? How might the architecture change if this requirement is removed?

2. How many architectural styles can you recognize in the architecture for ISSS?