
DEFINING THE RIGHT REQUIREMENTS: An Avoidable Pitfall in Rehosting Legacy Ground Systems to Modern Architectures

H. M. Sielski, C. T. Knapp, and R. D. Pendley

*Federal Sector - Defense Group
Aerospace Information Technologies
Computer Sciences Corporation
Rockville, Maryland*

Agenda

- **Ongoing and recent legacy rehosts**
- **Ideal requirements development**
- **Requirements development under stress**
- **An illustrative example**
- **A few recommendations**

Satellite Ground Systems Never Die...

...but they do get replaced.

- **NASA**
 - Complete rehost of all flight dynamics systems at Goddard Space Flight Center under cNMOS program
 - Consolidating control systems into IMOC under CSOC programs
- **NOAA**
 - Replacing GIMTACS with COTS system
- **USAF**
 - Replacing of CCS for MILSATCOM under MISCS
- **INTELSAT**
 - Replacing flight dynamics and commanding systems under FDC program

No doubt there are others...

How is all this Updating Going?

There are some setbacks:

- Schedules generally stretch out
- Costs escalate
- Replacement system performance lags the legacy system

...much the same as original development

These setbacks are surprising because:

- Known functionality - the legacy system is flying spacecraft
- Modern ground system COTS products are getting better
 - growing user base
 - increasing adaptability
- Modern distributed systems have significant processing power and flexibility

As with all complex problems, there are many factors, but the focus here is on *requirements development*.

Ideal Requirements Development Process

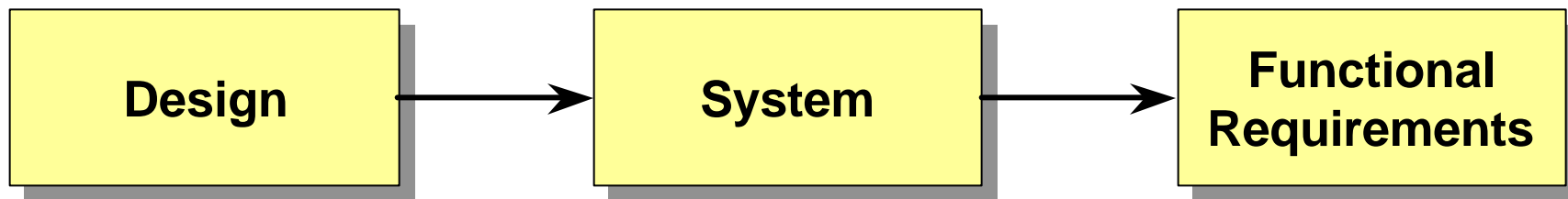


- **Operational requirements and Concepts of Operations are the basis of functional requirements**
- **As the CONOPS evolves and additional functionality is added, the functional requirements are updated**
- **Design proceeds from functional requirements**
 - reuse considered after functional requirements are defined
 - designers interact with engineers to validate design
- **System qualified against well defined functional requirements**

Actual Requirements Development Process

Schedule and cost pressure often force design and development to proceed with functional requirements documented later (if ever).

- Proposed reuse often imposes a design constraint that in turn dictates functional requirements
- Requirements derived from design force a rehost designer back to the original design
 - generally a poor fit for a new operating system and COTS products
 - seldom account for additional functionality added in operations



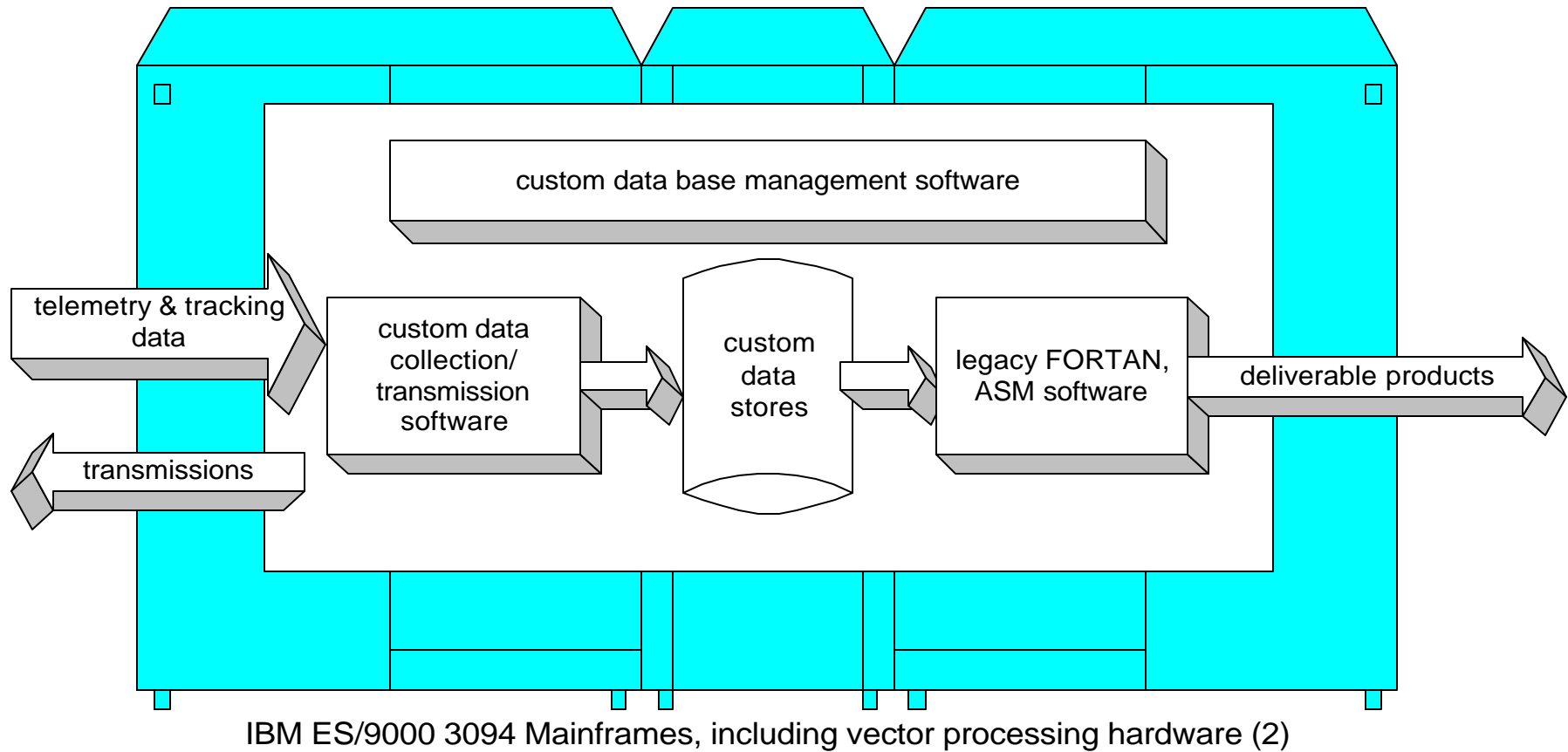
An Illustrative Example

NASA Goddard Flight Dynamics Facility (FDF) Mainframe Transition

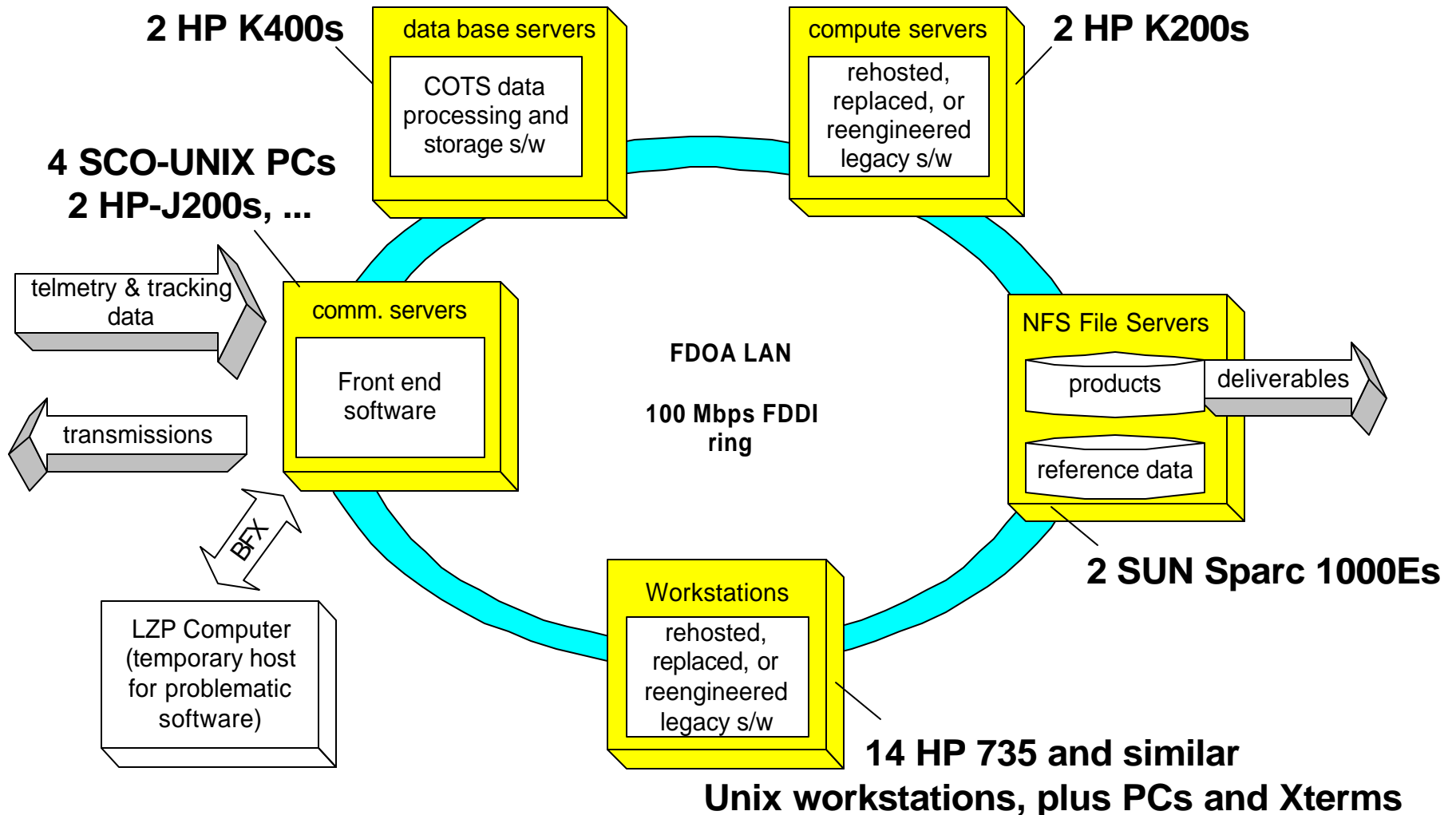
- **Single, overriding systems engineering objective:**
 - get off of the mainframe, as soon as possible, with no customer impacts
- **Drove requirements definition process:**
 - operations brought in, and told to identify everything that was needed currently to provide zero impact to customers
 - upcoming missions' system engineers brought in, and told to identify everything they needed to provide zero impact to customers
 - these became the high-level requirements
 - all other “desirements” discarded

High level requirements derived only from mission needs

FDF System Data Flow Before Transition...



... And After Mainframe Transition



Transition System Engineering Accomplishments

- **External customers unaffected, unless they desired a change**
- **Got off of the mainframe without impact to (or loss of), unmanned spacecraft and launch vehicles, and without impact to Human Space Flight**
- **All interfaces encapsulated, creating architecture that could be broken down, and moved out of institutional facility if necessary**
- **Requirements on the system tied to external customers, allowing for retirement of requirements and systems when external missions no longer desired support**

Approximate Before-and-After Statistics

| | Before Transition | After Transition |
|---------------|-------------------|------------------|
| Unix or PC | 1.5 MSLOC | 3.5 MSLOC |
| FDF mainframe | 3.3 MSLOC | |
| LZP mainframe | | 0.2 MSLOC |
| Totals | 4.8 MSLOC | 3.7 MSLOC |

2.8 MSLOC rehosted, reengineered, or replaced
0.3 MSLOC functionality retired
0.2 MSLOC rehosted to LZP computer

23 months, 120 staff years, all contributors, including parallel operations

Use of COTS (and COTS-like) Products

Before transition

- RIM
- IMSL
- IBM-specific tools such as ISPF, FBR/ADR, etc.

COTS use dominated by libraries, development tools, IBM mainframe infrastructure tools

After transition

- Oracle
- Omega
- Matlab
- IMSL
- Tcl/Tk
- ADSM
- Reel Exchange

COTS use dominated by RDBMS environment, specialty applications and languages, open system infrastructure tools

Some Recommendations (1 of 2)

- **Don't assume that:**
 - everybody knows what it does
 - every design requirement of current system is a necessary carryover
- **Invest in high-level system requirements definition at project outset:**
 - need strong driver for system end-state (e.g., only customer key interface requirements);
 - don't worry about subsystem allocation until high-level requirements are settled.

Some Recommendations (2 of 2)

- **Produce cost/schedule cutoff thresholds for “desirements:”**
 - allow for meaningful trades;
 - gateway for deleting functional requirements no longer needed; or for
 - separate design requirements (i.e., constraints) of legacy system implementation from functional requirements.