Kennedy Space Center Launch Processing System (LPS) and the Test Checkout and Monitoring System 2 (TCMS2)

Robert Luken, NASA/KSC

Evolution
of
LPs

A Lukenfilm
presentation

Topics

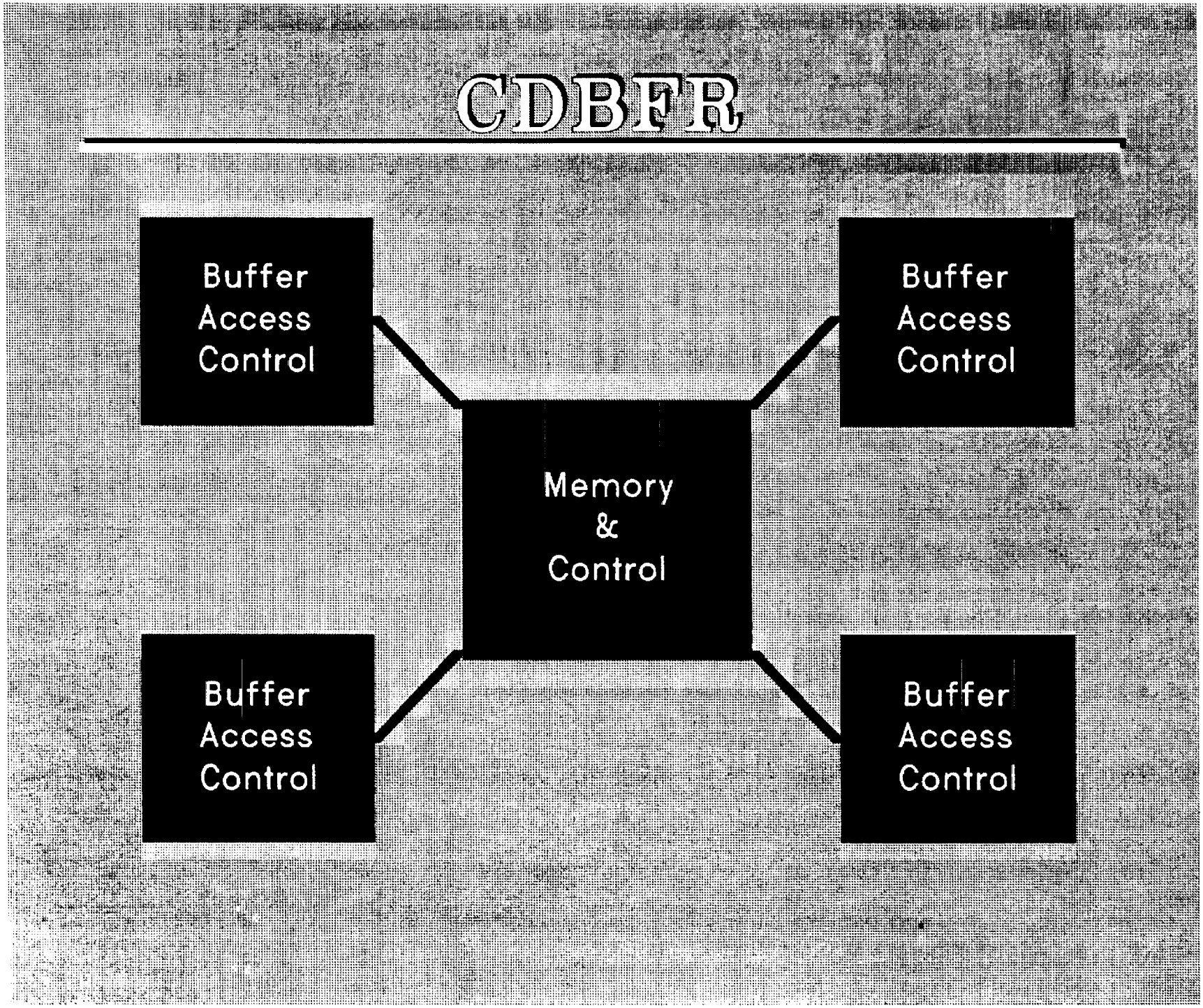
- Review of KSC/LPS
- Status of Development efforts
- Preview of LPS II
- Lessons Learned

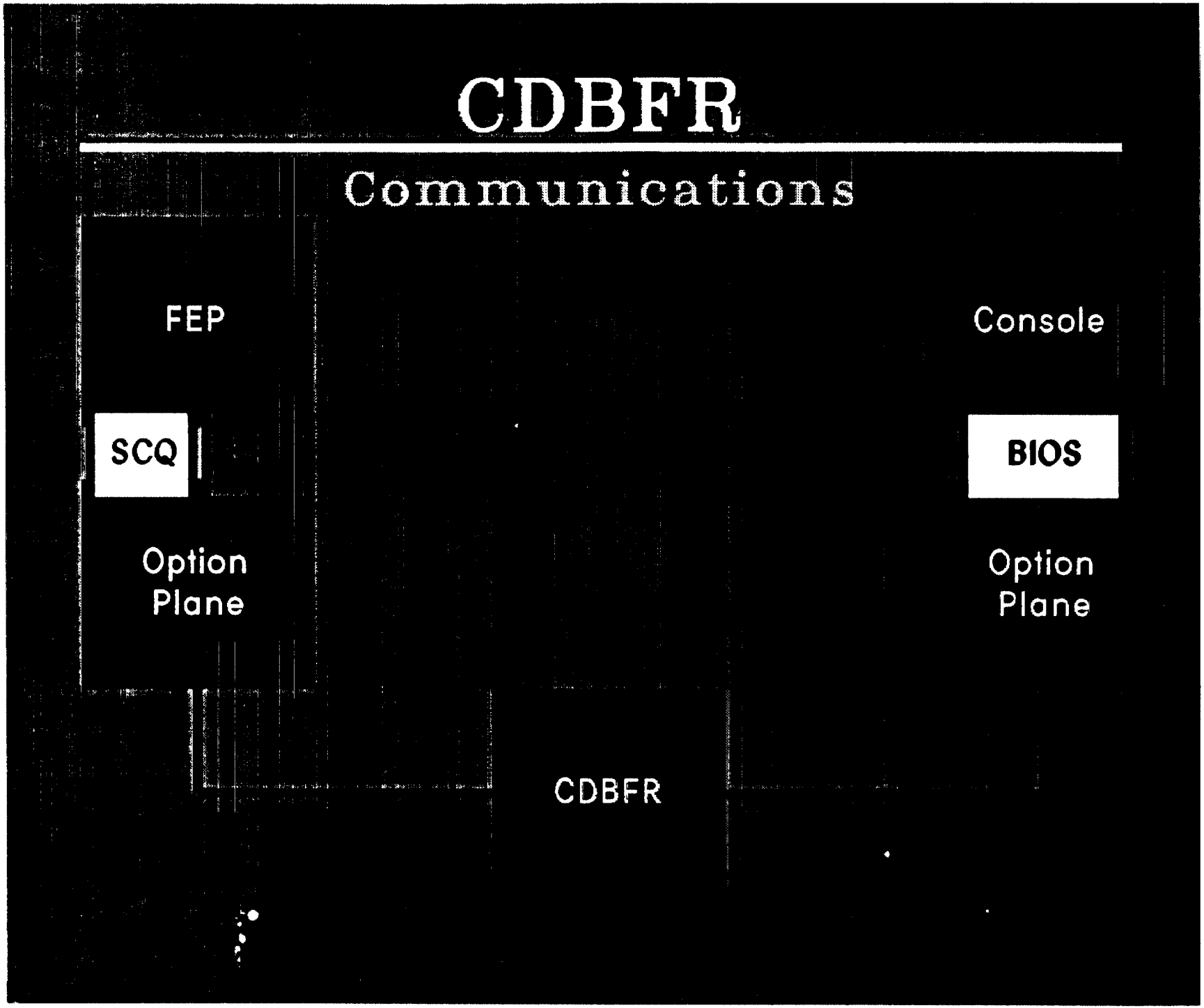
Historical Review

LPS Milestones

Conception thru Operation

LPS Architecture
Front End Processor
Hardware Interface Module
Front End Processor
Processed Data Recorder
Console
Common Data Buffer
Shared Peripheral Area
Console


CDBFR
Buffer Access Control
Buffer Access Control
Memory & Control
Buffer Access Control
Buffer Access Control


CDBFR
Communications
FEP
Console
SCQ
BIOS
Option
Plane
Option
Plane
CDBFR

LPS Architecture

Pros

- Modular Design
- Star Architecture
- High Throughput-Low Latency
  - ~ 1 msec GOAL Response
  - ~ 5 msec CTC Response
  - ~ 37 msec Reactive Loop Response
- Semi-Demand Mode of Operation

LPS Architecture

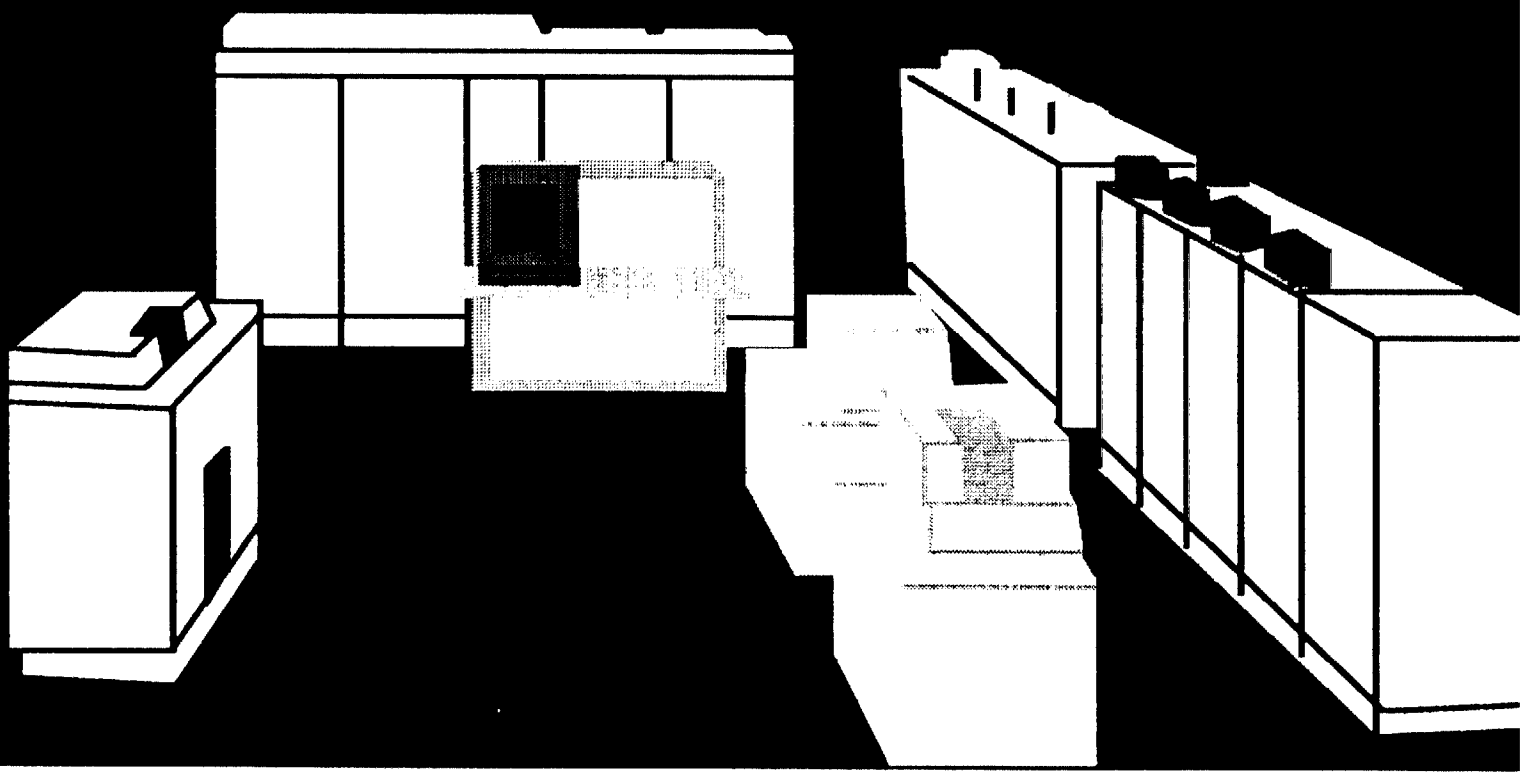
Cons

- Single Point Failure (CDBR)
- Independent Sets
- 64 Port 64Kword (Implementation)
- Recording Bottleneck
- Not Partitionable
- Compile Time Bindings
- Dynamic Buffer Map

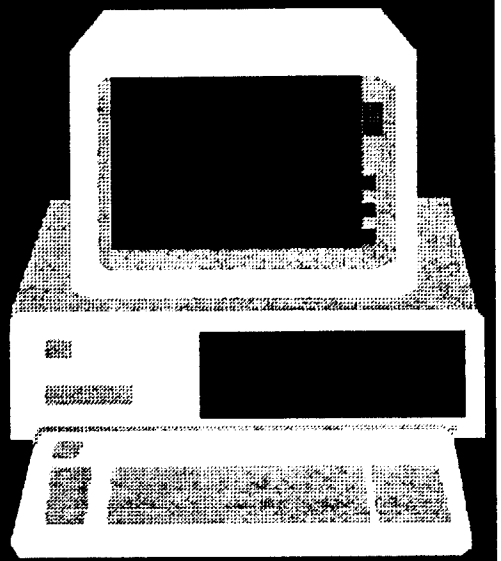
Future Systems

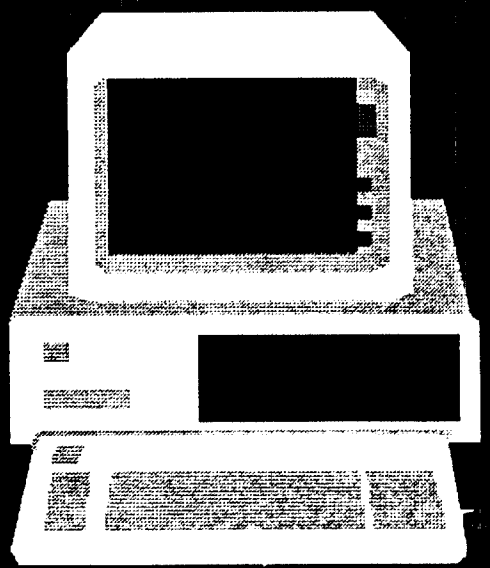
Centralized Approach

- Easier development
- Easier Management



Distributed Approach





- Expandability
- Flexibility
- Modularity
- Reliability

Goals

Flexibility

✓ Multiple application languages

C

GOAL

USE

Ada

Common Lisp

Pascal

Fortran

✓ Reconfigurable for user needs

✓ Support for custom interfaces

✓ Vendor independent

Goals

Modularity

✓ Building block approach

✓ Hardware & software independence

✓ Isolate common functions

Goals

Maintainability

- ✓ High degree of system health checking
- ✓ Subsystem diagnostics to LRU level
- ✓ Modules brought on & off line without impact
- ✓ Minimum number of unique LRUs

Goals

Compatibility

- ✓ Use industry standard protocols
- ✓ Use industry standard interfaces
- ✓ Use standard data interchange formats
- ✓ Use industry standard operating system

Goals

Reliability

- ✓ Distributed architecture minimizes failure effects
- ✓ High MTBF equipment specified
- ✓ FO/FS configurations supported

Goals

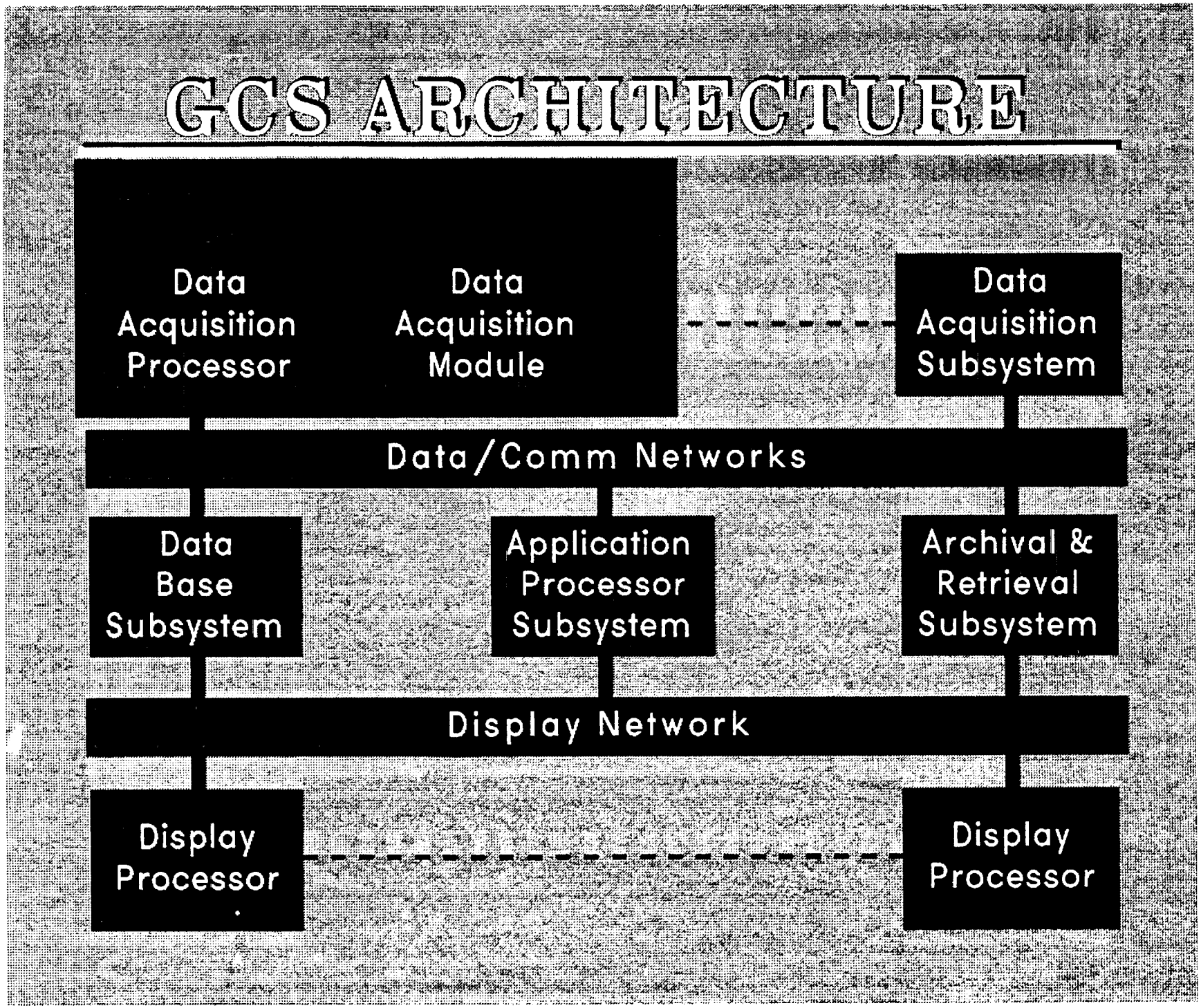
Upgradeability

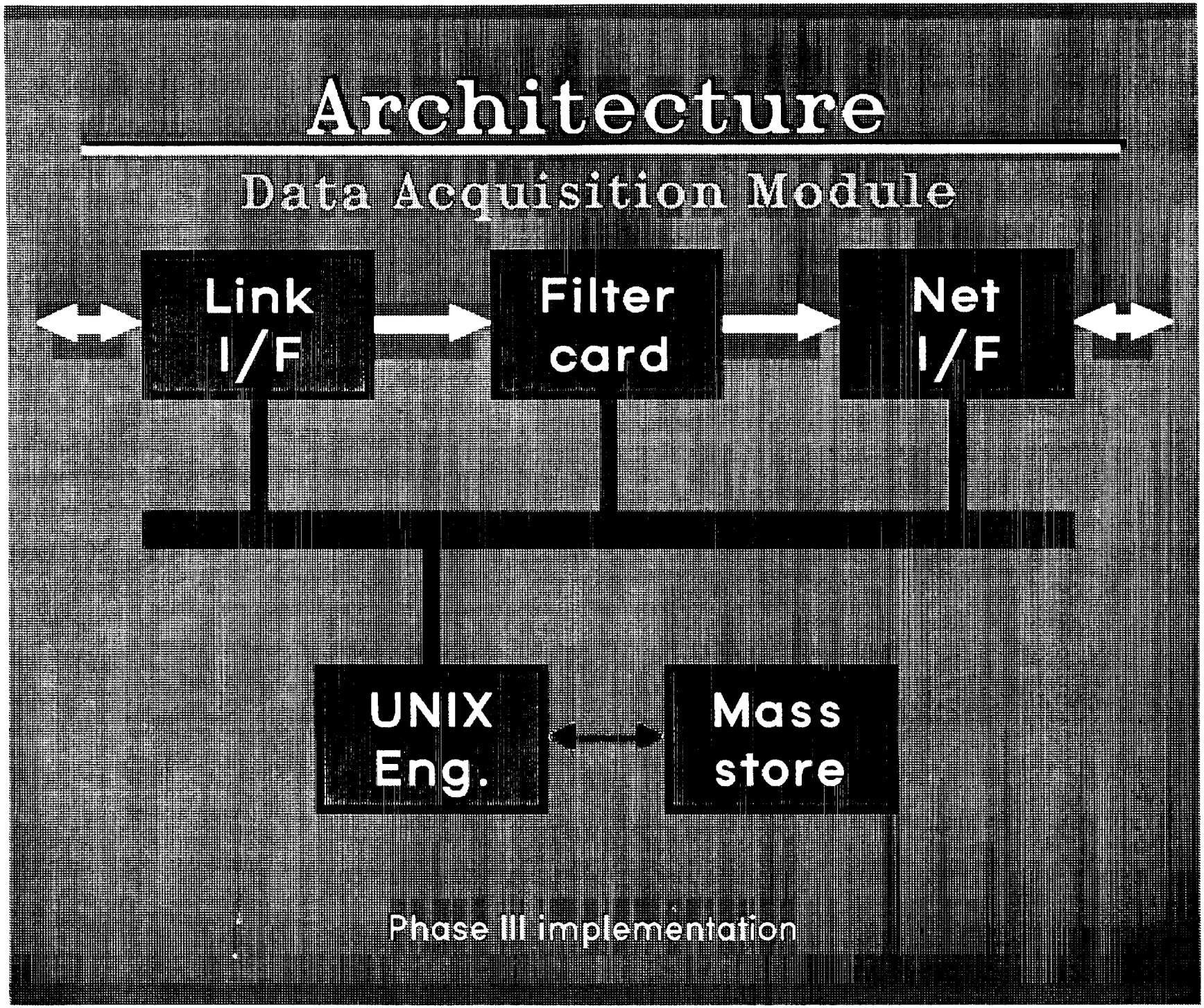
- ✓ Utilization of industry standards
- ✓ Vendor independence
- ✓ Modular implementation
- ✓ Hardware & software independence
- ✓ Planned migration path for upgrades

Goals

Affordability

- ✓ Commercial H/W & S/W
- ✓ Multiple vendors
- ✓ Minimize use of unique H/W & S/W


GCS ARCHITECTURE
Data Acquisition Processor
Data Acquisition Module
Data Acquisition Subsystem
Data/Comm Networks
Data Base Subsystem
Application Processor Subsystem
Archival & Retrieval Subsystem
Display Network
Display Processor
Display Processor


Architecture
Data Acquisition Module
Link I/F
Filter card
Net I/F
UNIX Eng.
Mass store
Phase III implementation

GCS Architectures

Pros

- Modular Design
- Standard Interfaces
- Primarily COTS
- Partionable
- Connectivity
- Run Time Binding
- Low Latency Data Reads

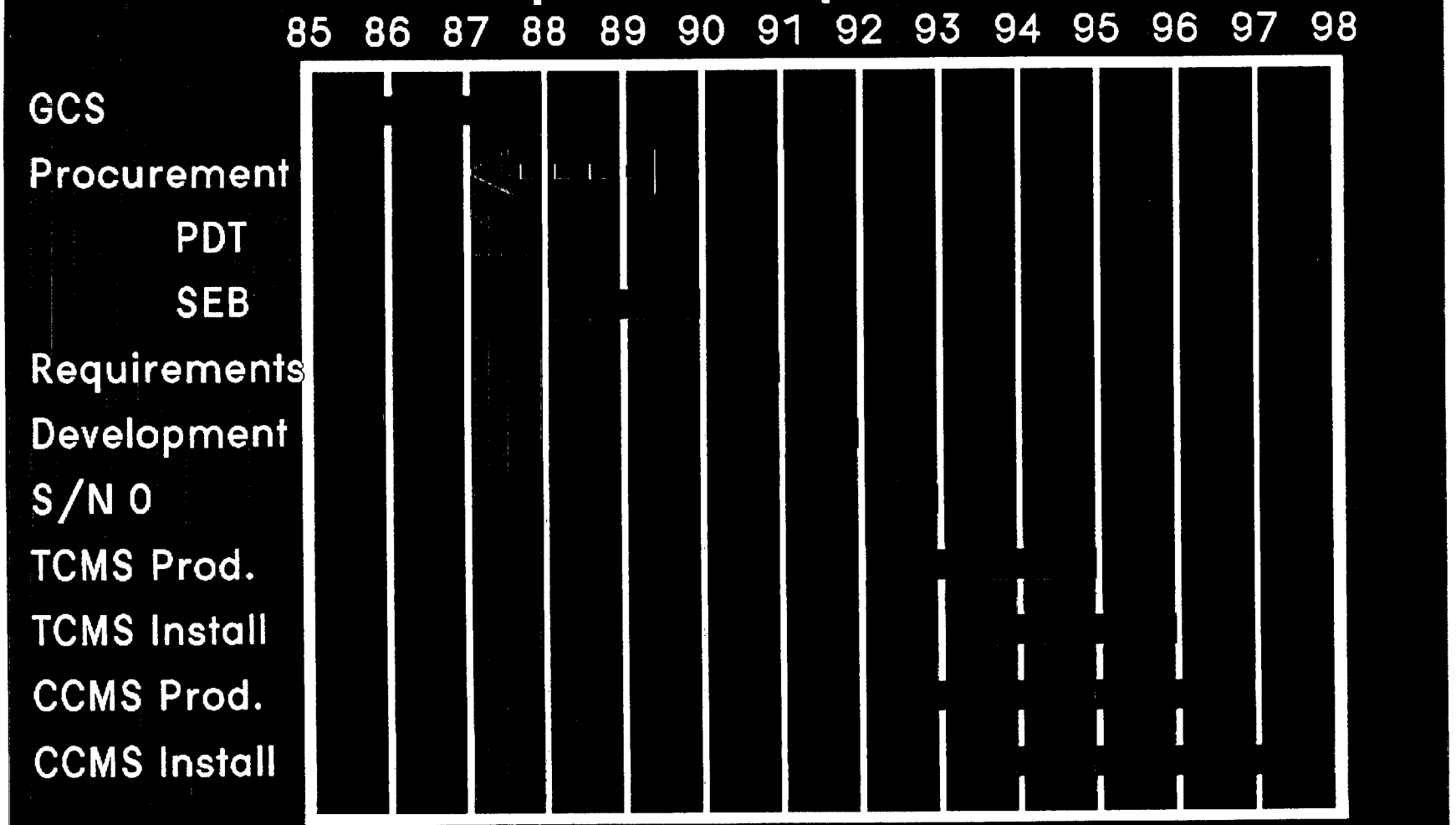
GCS Architecture

Cons

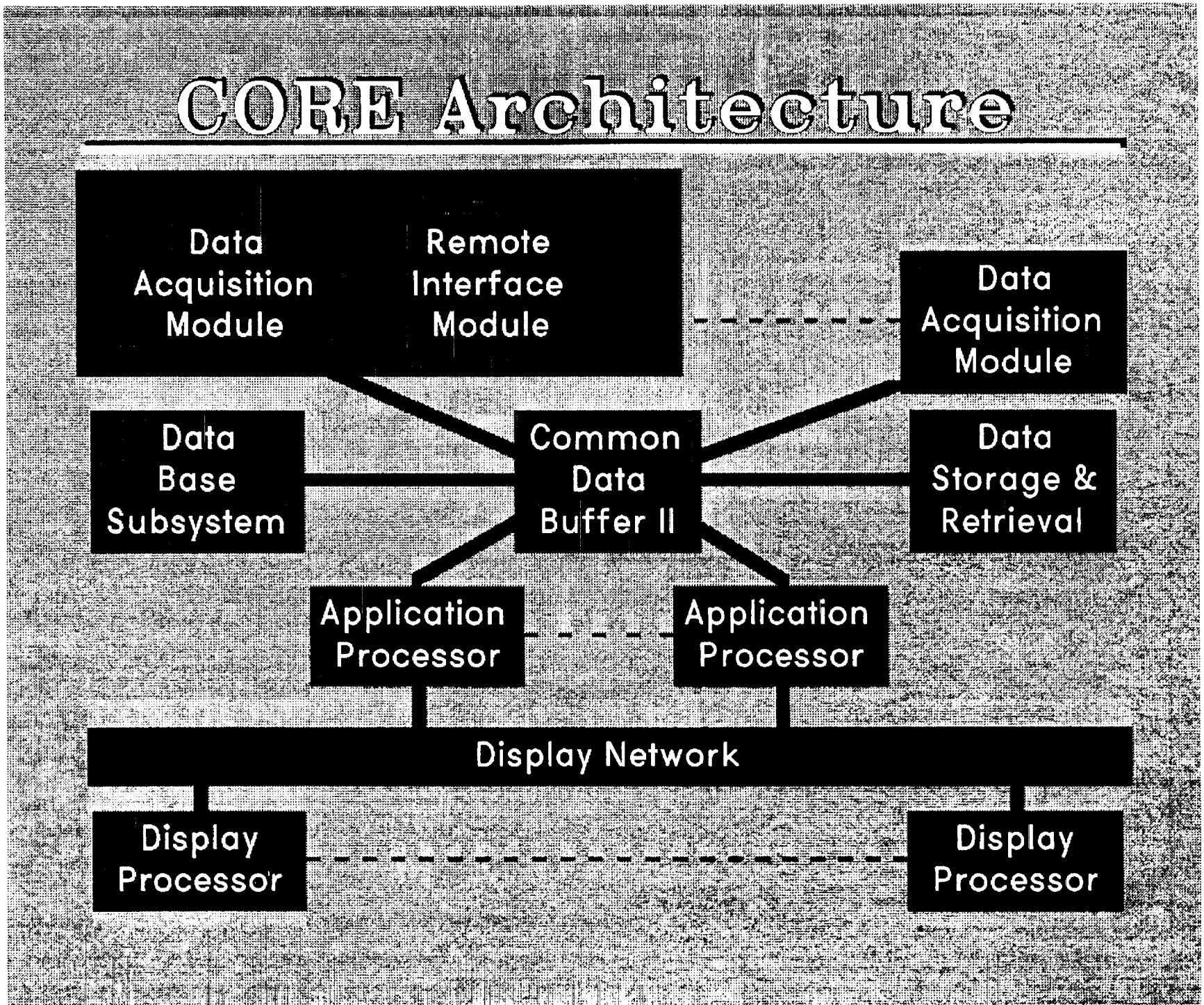
- Broadcast Mode of Operation
- Recording Bottleneck
- Excessive MIP Requirements
- High Overhead Data Format
- CDS/RTIF Incompatibility
- CTC/Reactive Loops Slow
- Weak Simulation Support

LPS II Milestones

Concept thru Operation



CORE Architecture



Lessons Learned

- ✓ NIH & DIY significant factors
- ✓ COTS not a perfect fit
- ✓ Almost COTS worst of all
- ✓ "It doesn't work that way now!"
- ✓ "Trained monkeys" can't maintain it
- ✓ Config. Mgmt. driving factor
- ✓ Development & Ops don't mix