System Architecture & Security

Experiences from real life design decisions

or

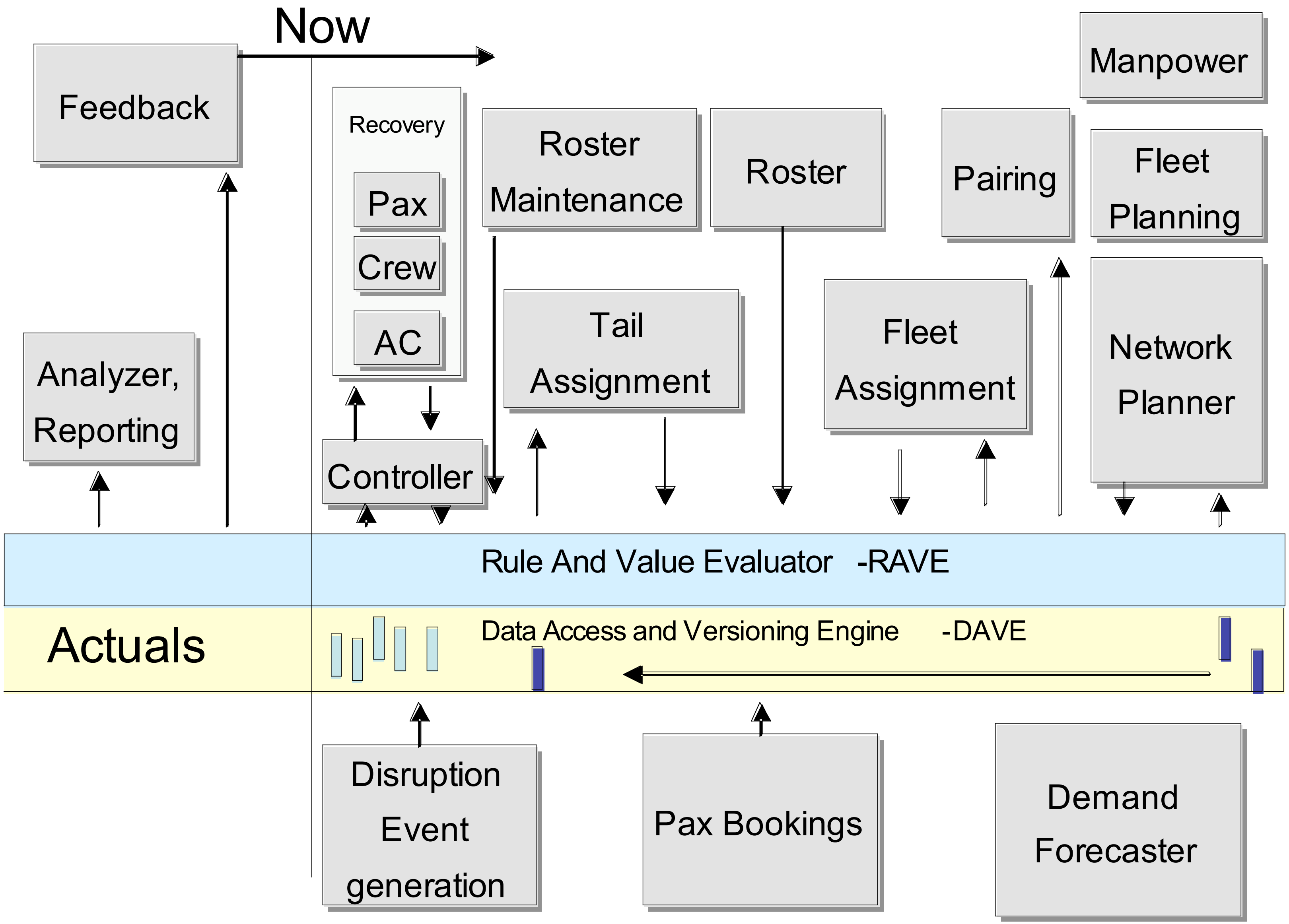
A network and security expert in API land

Martin Fredriksson
<m@carmensystems.com>

Contents

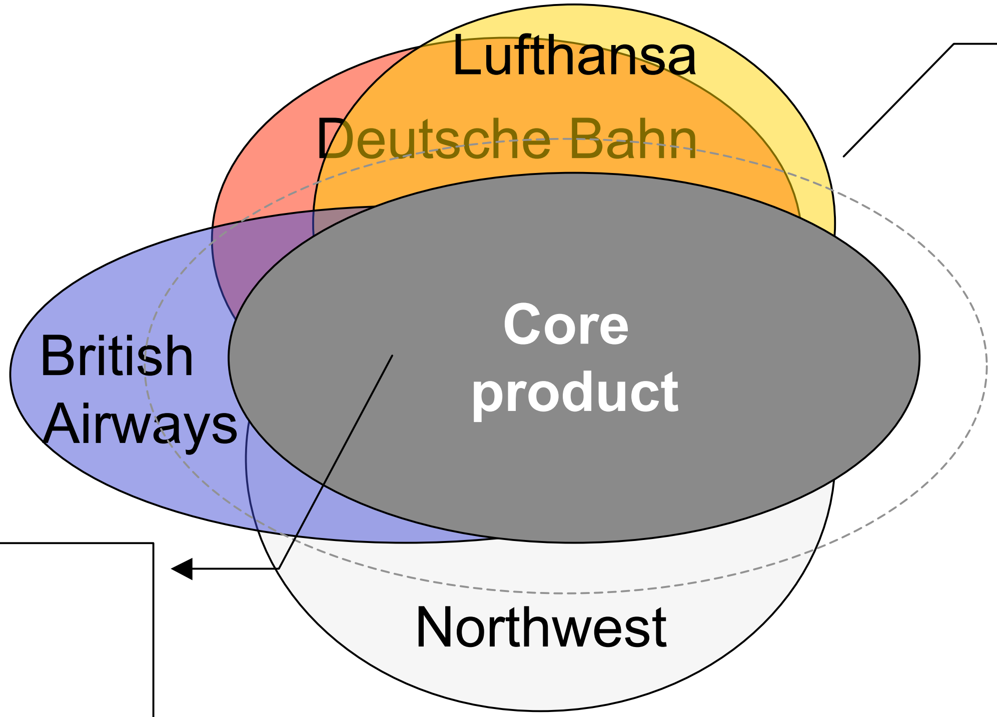
- *Introduction*: lightspeed overview of the Carmen system
- *We have assumed control*: System Architecture views
- *Steps in the right direction*: Design examples and discussions

Airline-in-a-box



CARMEN SYSTEMS
RESOURCES IN BALANCE

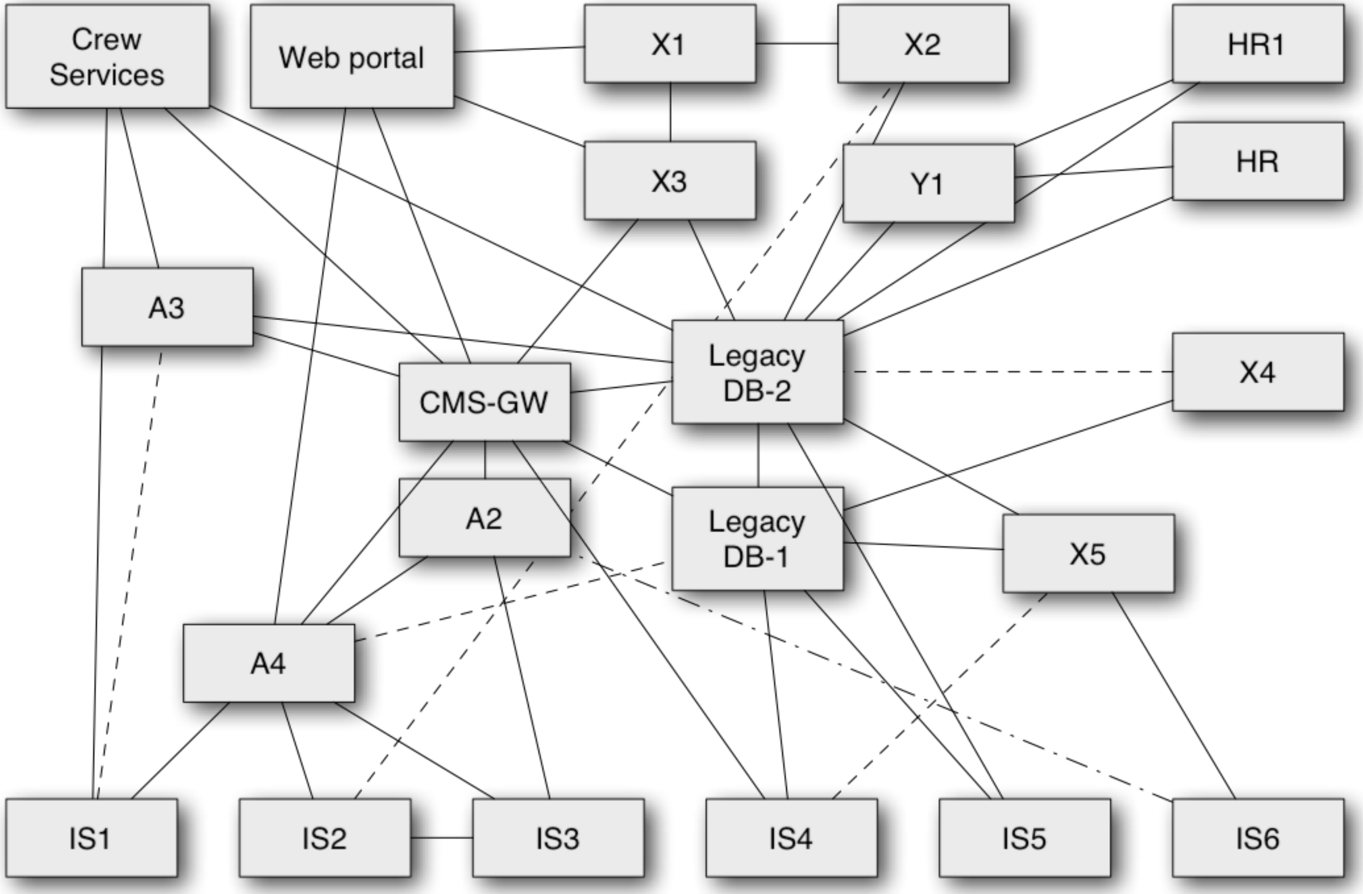
Carmen development/business model



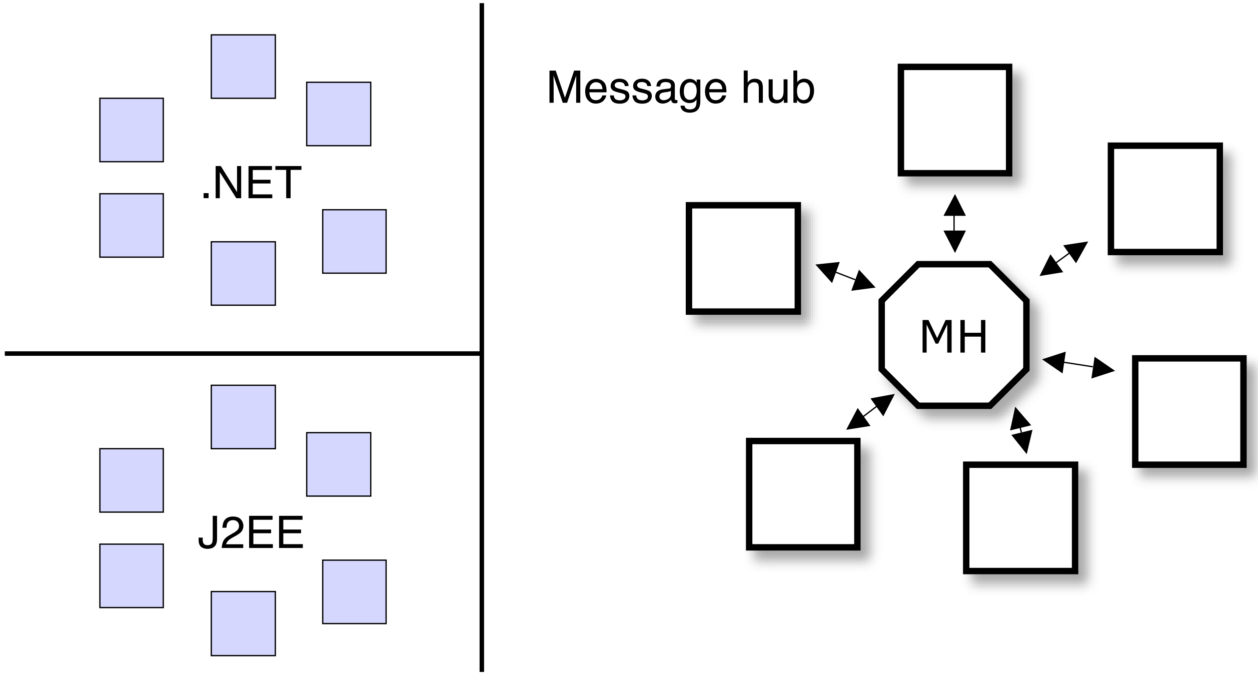
- Legality
- Quality
- Costs
- Reports
- Interfaces
- Colours
- etc...

- Optimization methods
- GUI
- Modelling tools like Rave and the report generator
- Basic functionality (like security)

Why Architecture?
What's wrong with this picture?



Star Architecture

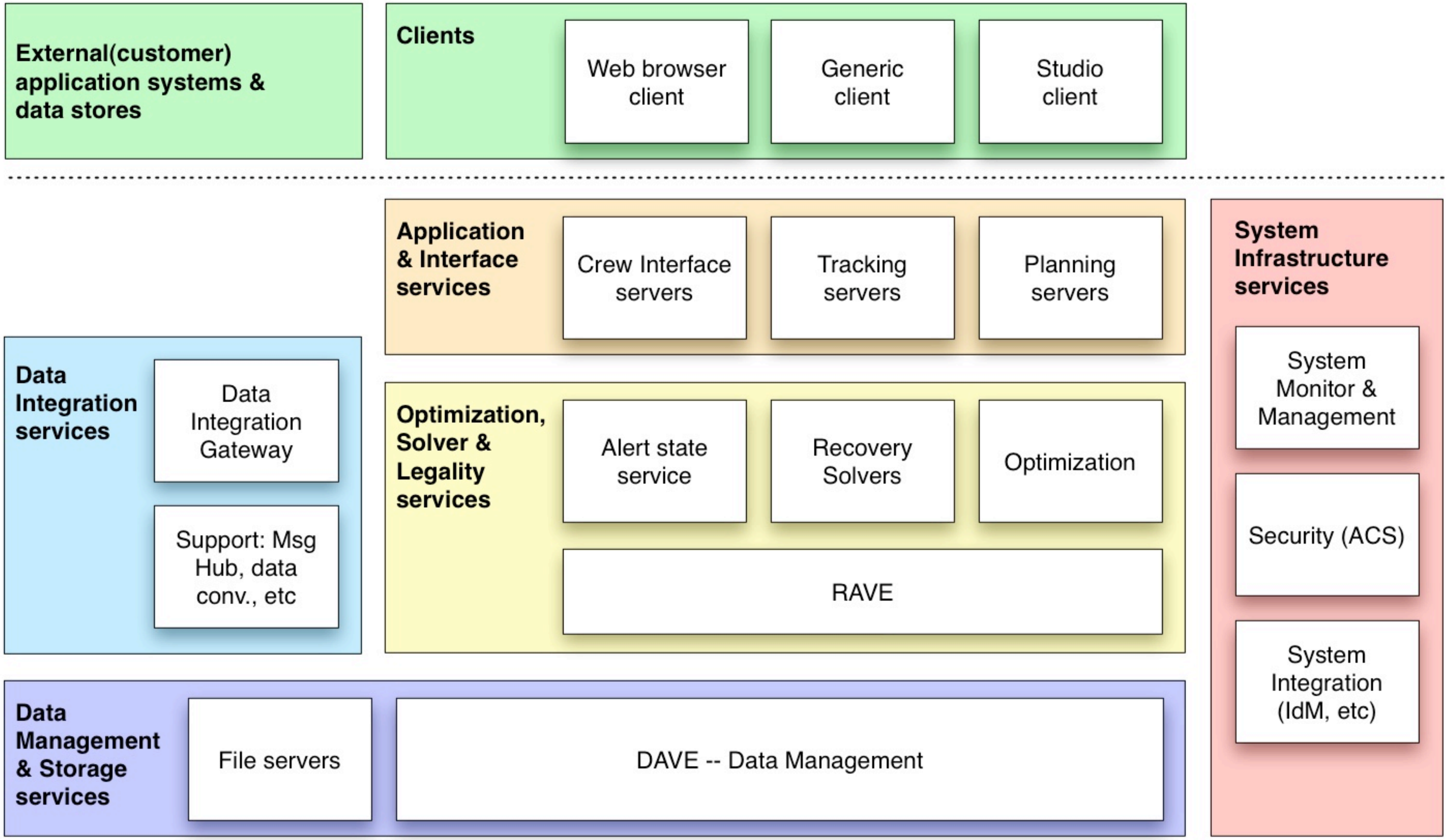


But what about:

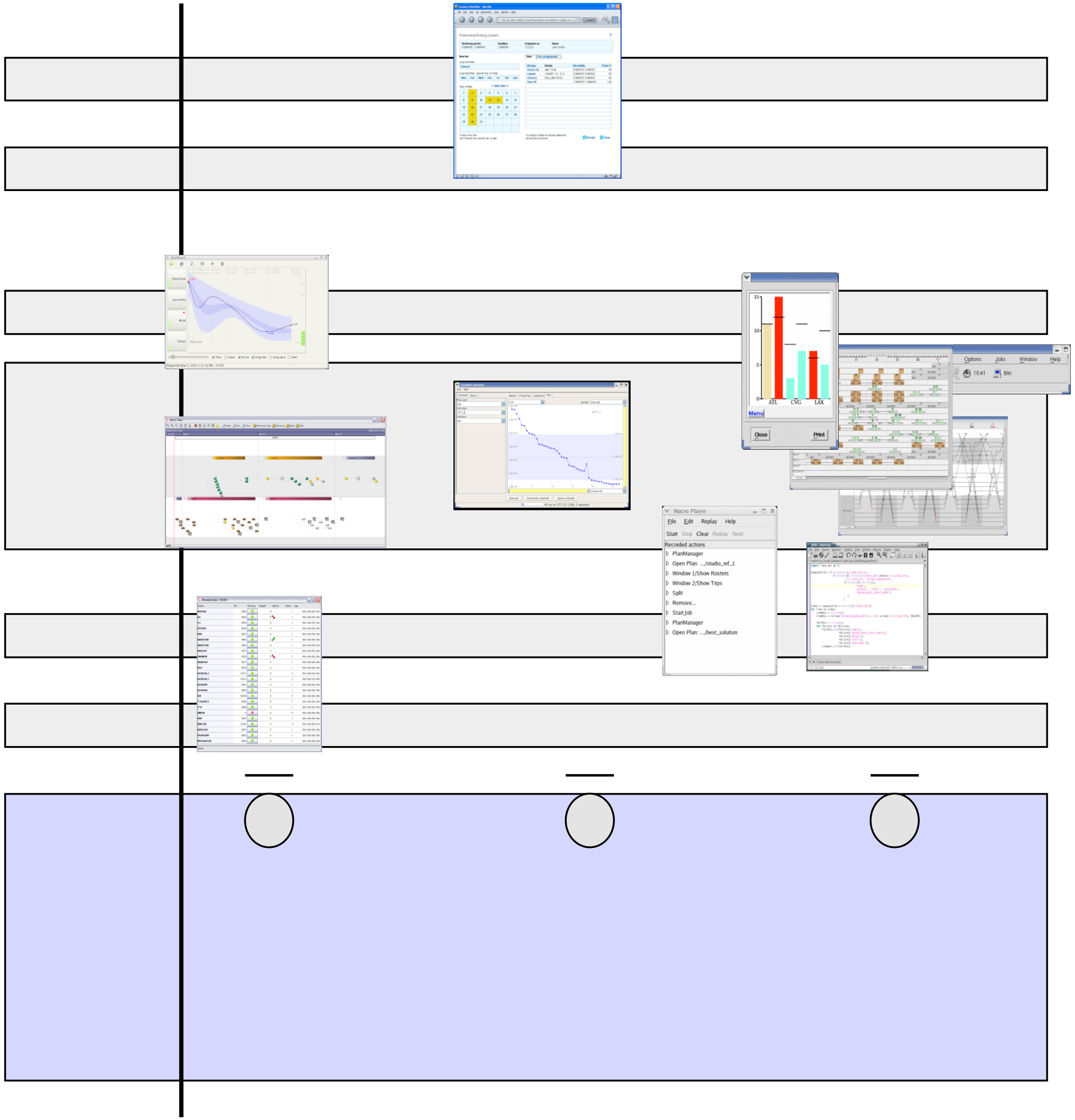
Synchronizing activities?
Auditing?
System maintenance?
Deployment?
Network security?

Not enough for a complete *System*…

System Architecture Overview


External(customer) application systems & data stores
Clients
Web browser client
Generic client
Studio client
Application & Interface services
Crew Interface servers
Tracking servers
Planning servers
System Infrastructure services
System Monitor & Management
Security (ACS)
System Integration (IdM, etc)
Data Integration services
Data Integration Gateway
Support: Msg Hub, data conv., etc
Optimization, Solver & Legality services
Alert state service
Recovery Solvers
Optimization
RAVE
Data Management & Storage services
File servers
DAVE -- Data Management

User Interfaces



Crew

Others

Portals

Managers

Dashboards

Planners

Trackers

Planboards

Reports

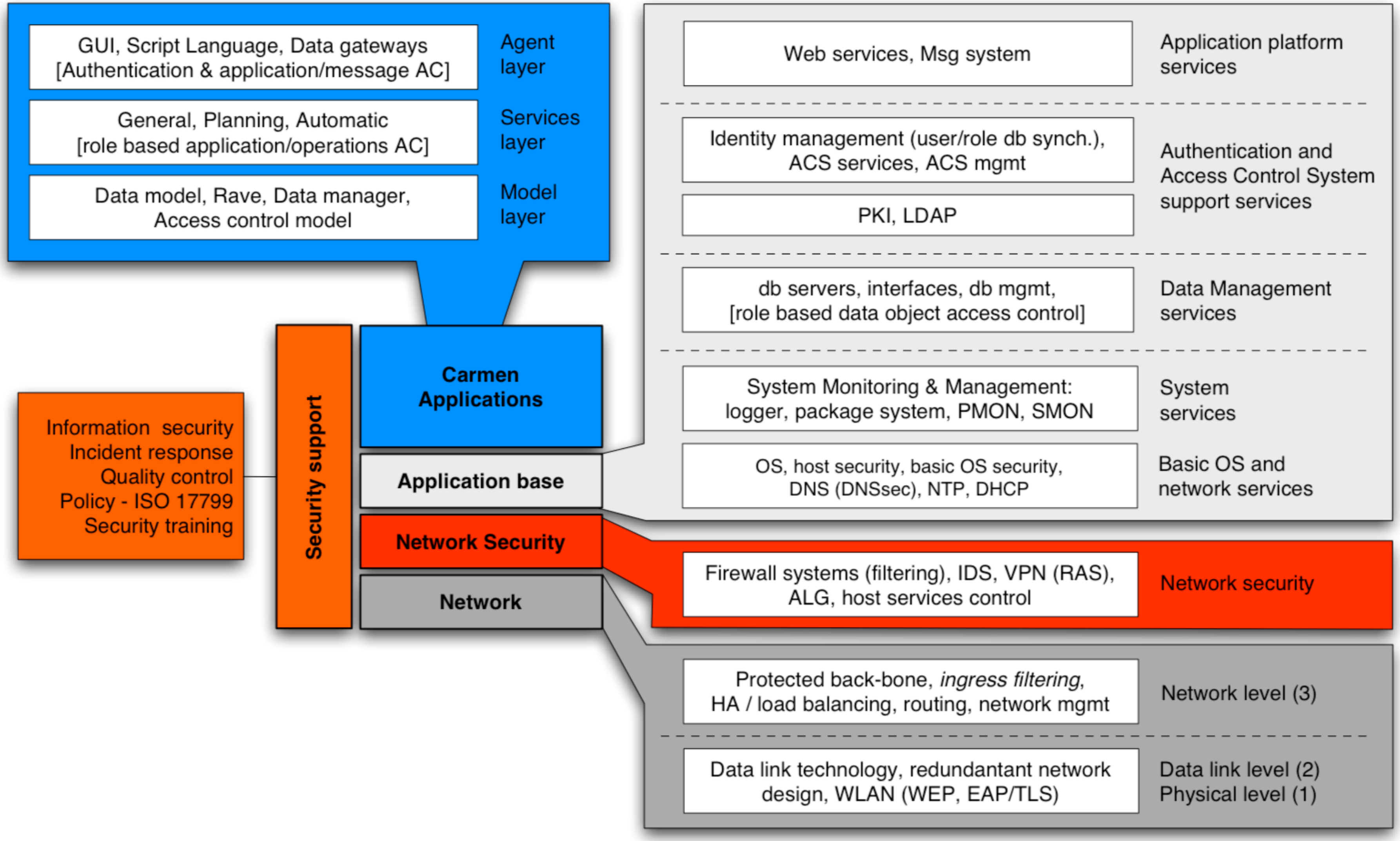
Engineers

Administrators

APIs & IDEs

Services

Layered System and Security Model



GUI, Script Language, Data gateways [Authentication & application/message AC] — Agent layer

General, Planning, Automatic [role based application/operations AC] — Services layer

Data model, Rave, Data manager, Access control model — Model layer

Information security
Incident response
Quality control
Policy - ISO 17799
Security training

Security support

Carmen Applications

Application base

Network Security

Network

Web services, Msg system — Application platform services

Identity management (user/role db synch.), ACS services, ACS mgmt — Authentication and Access Control System support services

PKI, LDAP

db servers, interfaces, db mgmt, [role based data object access control] — Data Management services

System Monitoring & Management: logger, package system, PMON, SMON — System services

OS, host security, basic OS security, DNS (DNSsec), NTP, DHCP — Basic OS and network services

Firewall systems (filtering), IDS, VPN (RAS), ALG, host services control — Network security

Protected back-bone, *ingress filtering*, HA / load balancing, routing, network mgmt — Network level (3)

Data link technology, redundantant network design, WLAN (WEP, EAP/TLS) — Data link level (2) Physical level (1)

Access Control Model

Key design principles:

Completeness: The ACS must always be invoked (e.g. used in all access methods) and impossible to bypass.

Isolation: The system must be tamper-proof.

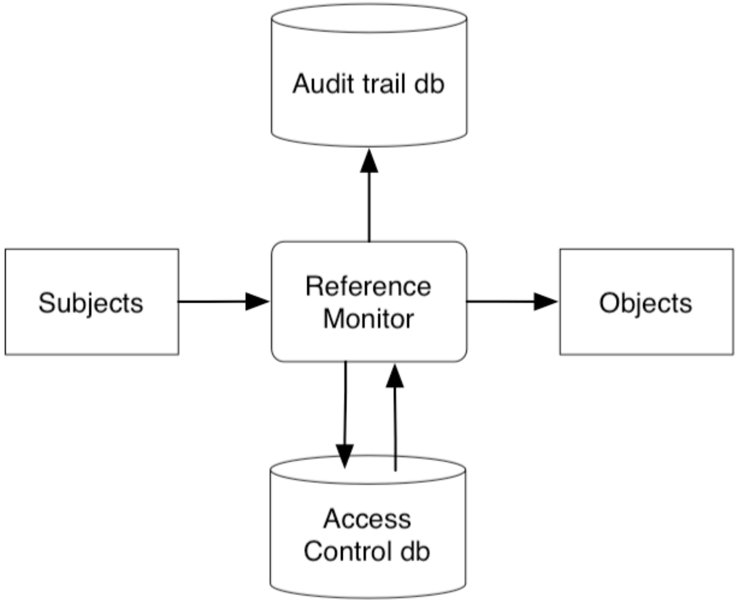
Verifiability: The system must be shown to be properly implemented.

Flexibility: The system should be able to enforce the access control policies defined by specific customer needs.

Simplicity: The security policy model should be kept as simple as possible (to minimize ACSdb and audit trail complexity) while still supporting the flexibility principle.

Manageability: The system must be easily manageable, via intuitive user interfaces using terms/rules that users can understand.

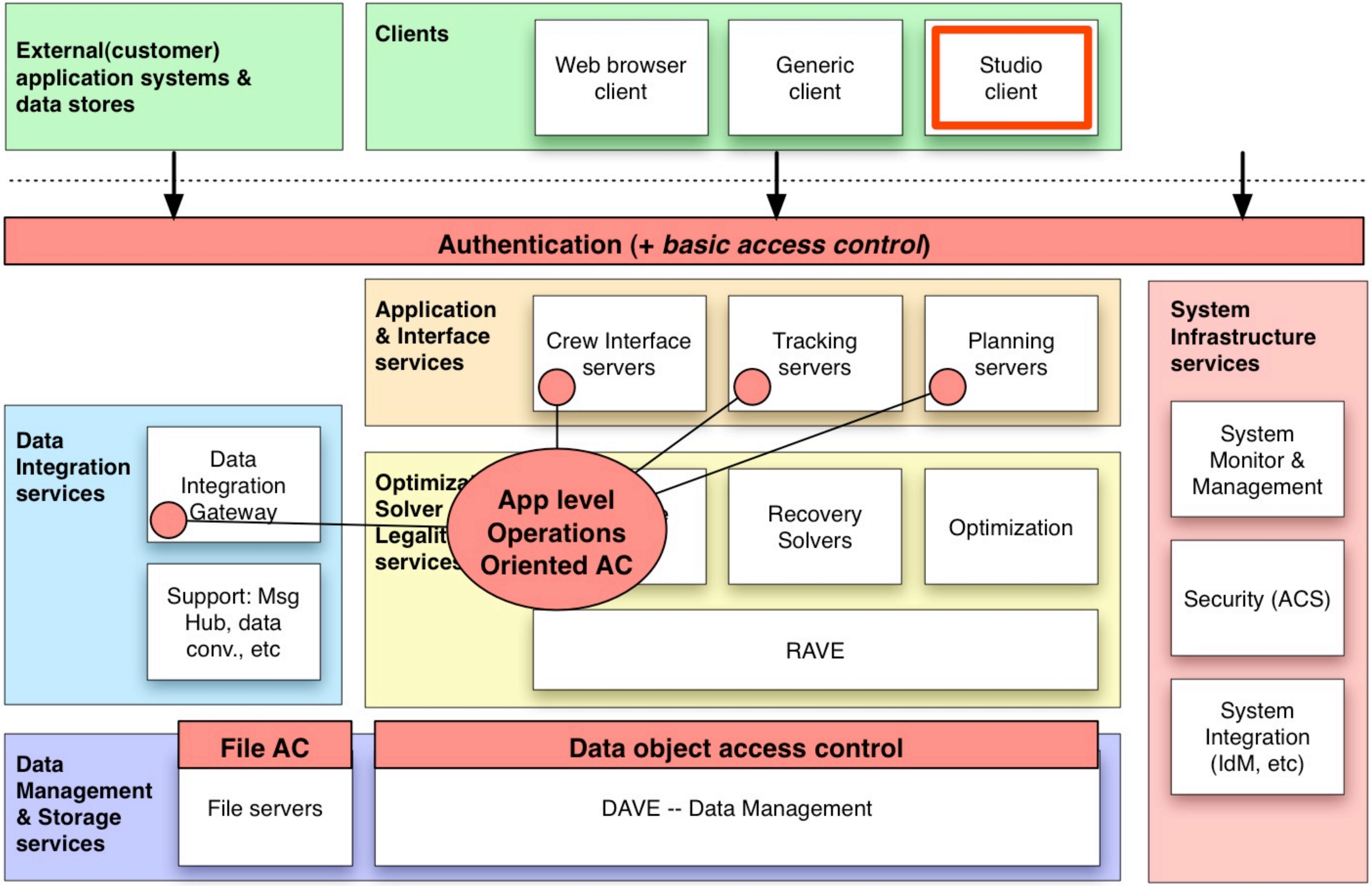
Scalability: The system must support the number of users/roles/data classes/resources needed by the customer.



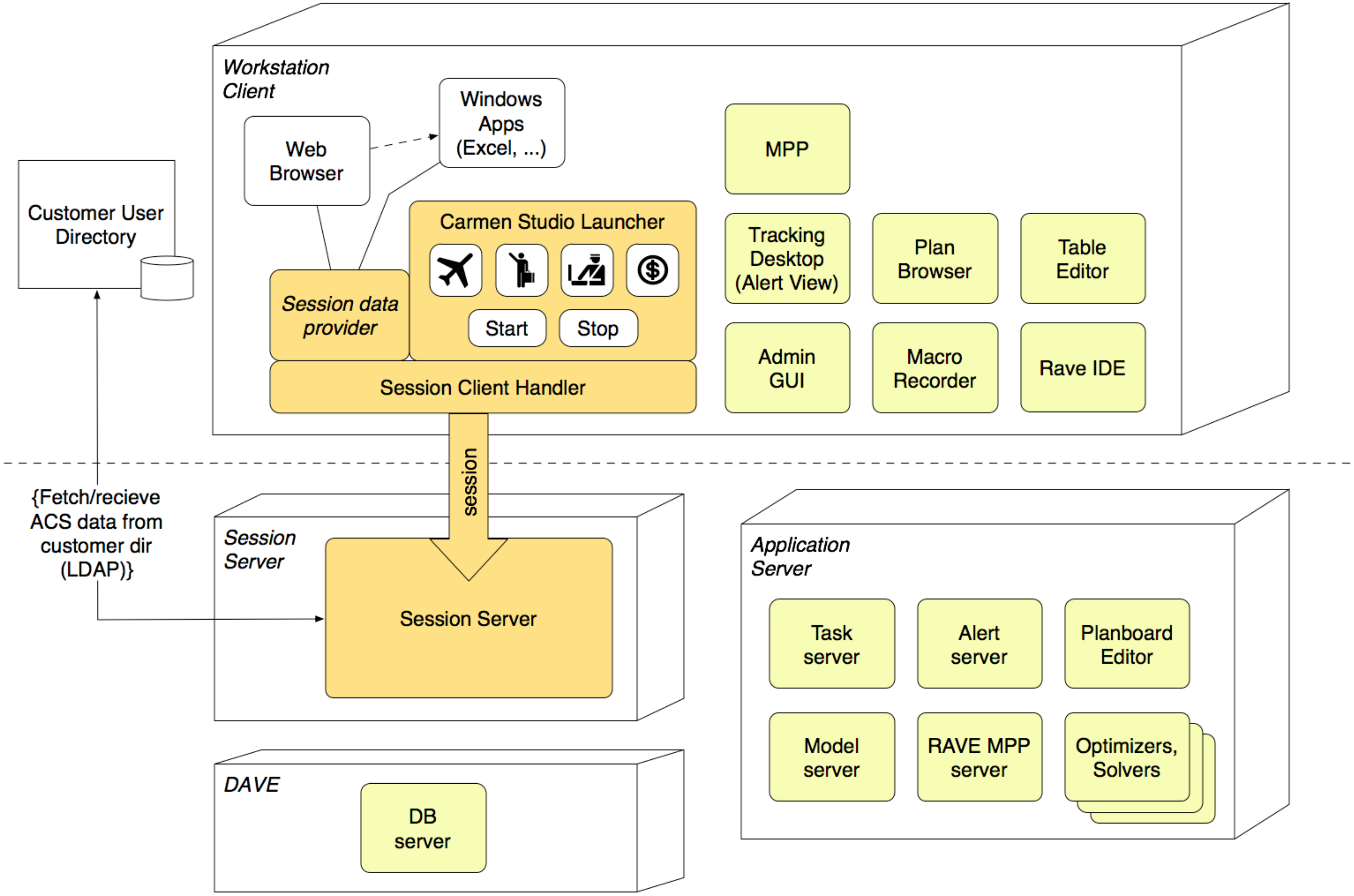
Reference Monitor Access Control model

All attempts by a subject to access an object are controlled by the reference monitor in accordance with a security policy embodied in the access control database. Security-relevant events are stored in the audit trail db (audit file).

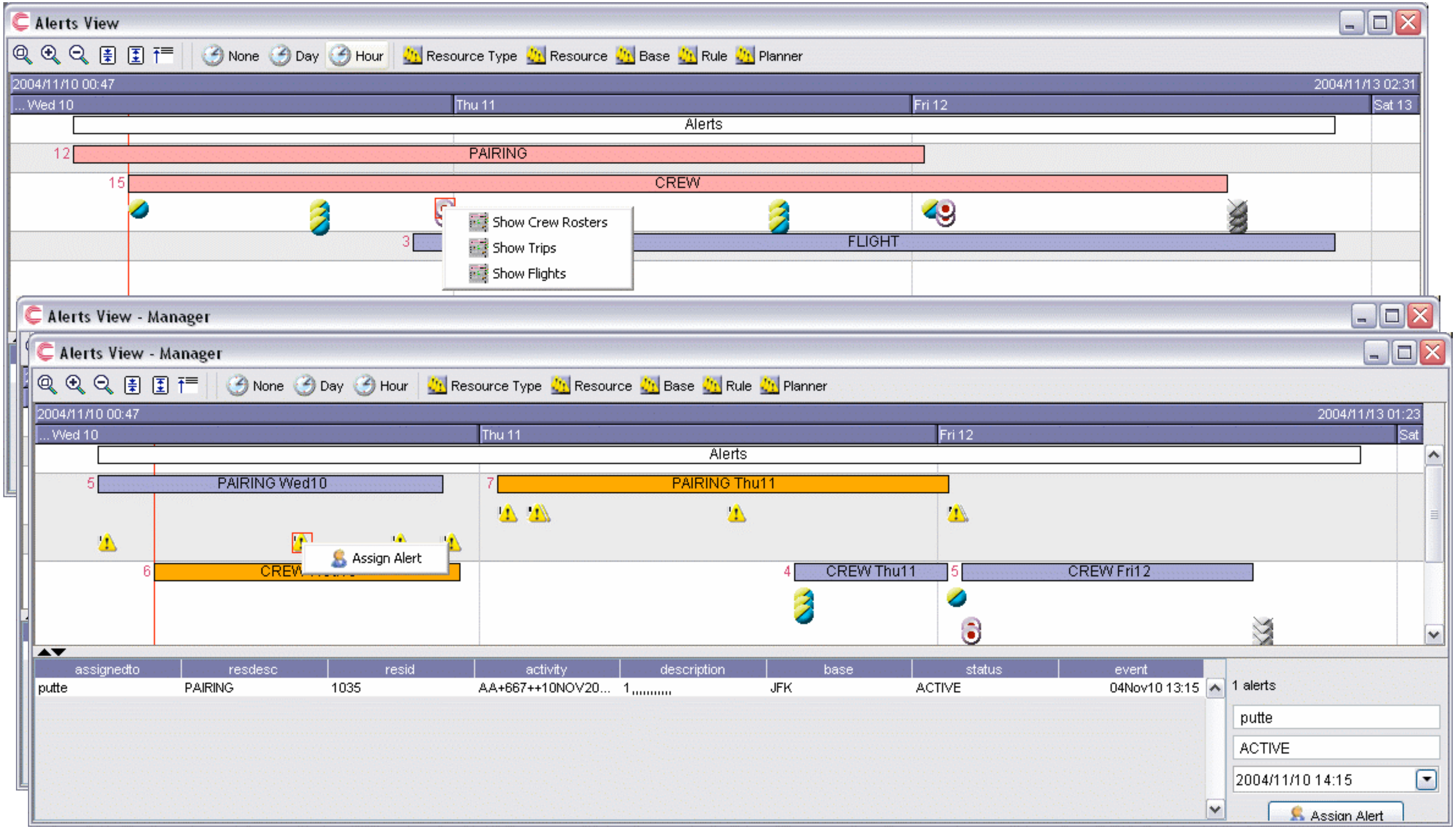
Three levels of the ACS


External(customer) application systems & data stores
Clients
Web browser client
Generic client
Studio client
Authentication (+ *basic access control*)
Application & Interface services
Crew Interface servers
Tracking servers
Planning servers
System Infrastructure services
Data Integration services
Data Integration Gateway
Support: Msg Hub, data conv., etc
Optimiza
Solver
Legalit
services
App level Operations Oriented AC
Recovery Solvers
Optimization
RAVE
System Monitor & Management
Security (ACS)
System Integration (IdM, etc)
Data Management & Storage services
File AC
File servers
Data object access control
DAVE -- Data Management

Client/Server Model



Different views for different *roles*



assignedto	resdesc	resid	activity	description	base	status	event
putte	PAIRING	1035	AA+667++10NOV20...	1,,,,,,,,,,,	JFK	ACTIVE	04Nov10 13:15

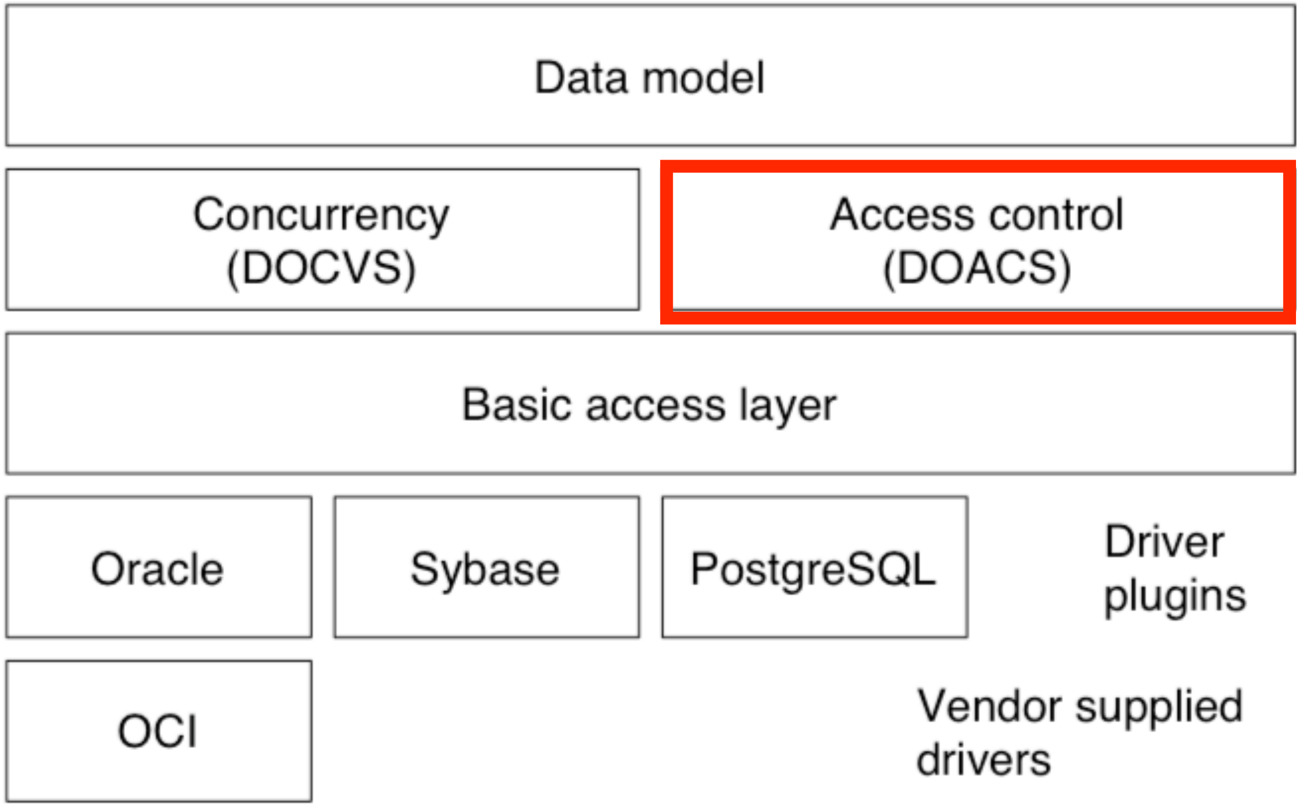
1 alerts

putte

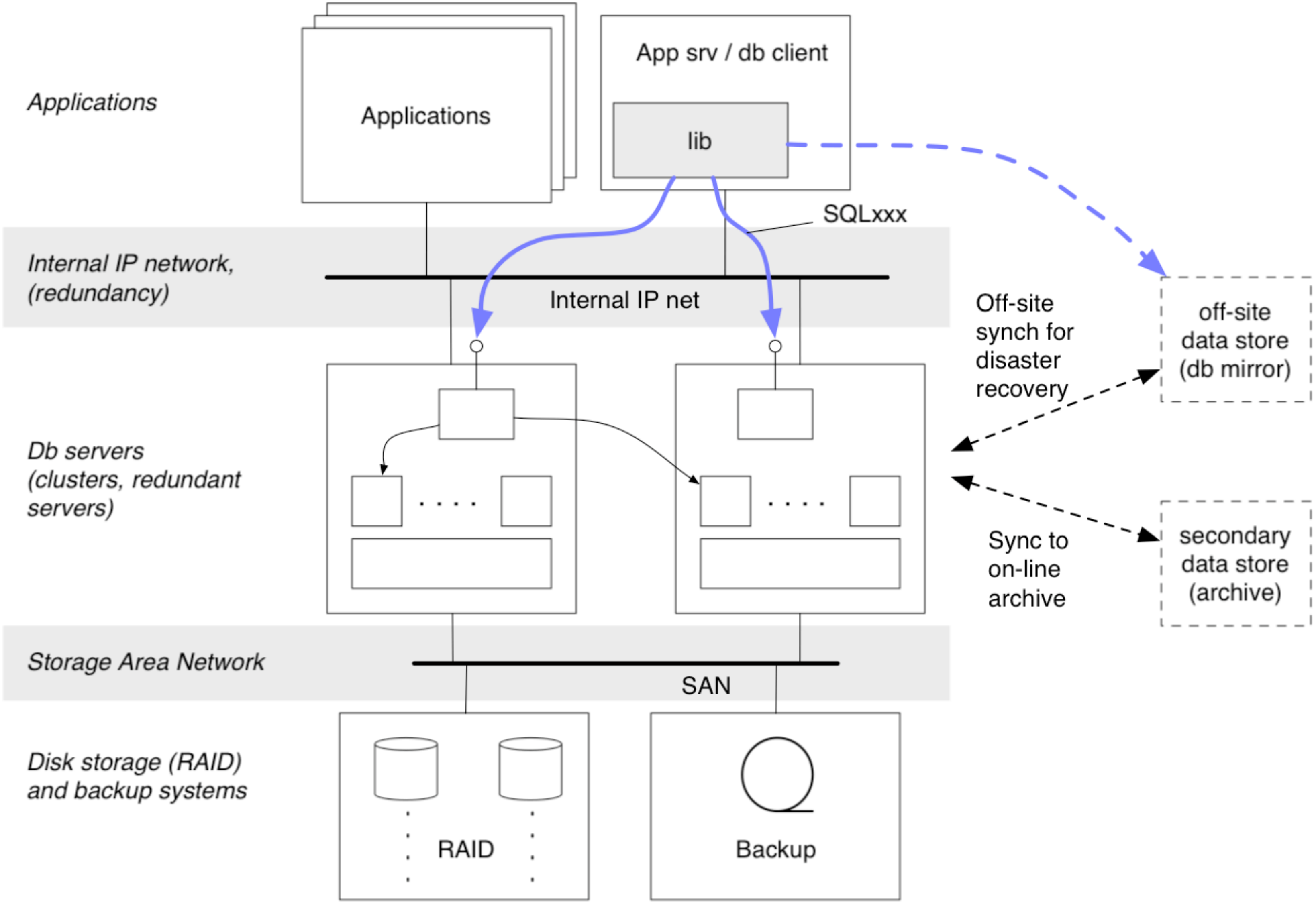
ACTIVE

2004/11/10 14:15

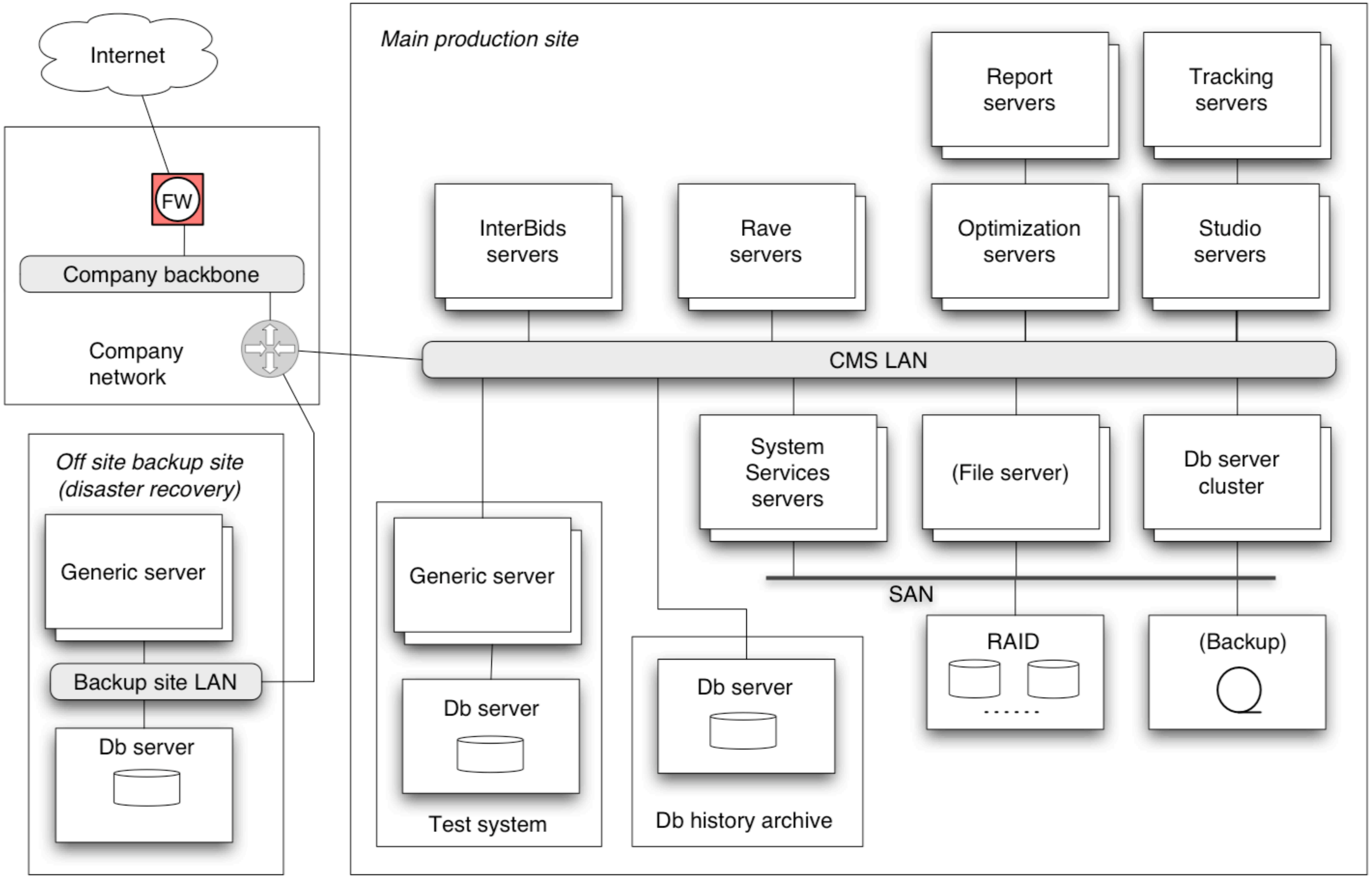
Data Management (DAVE) layered model



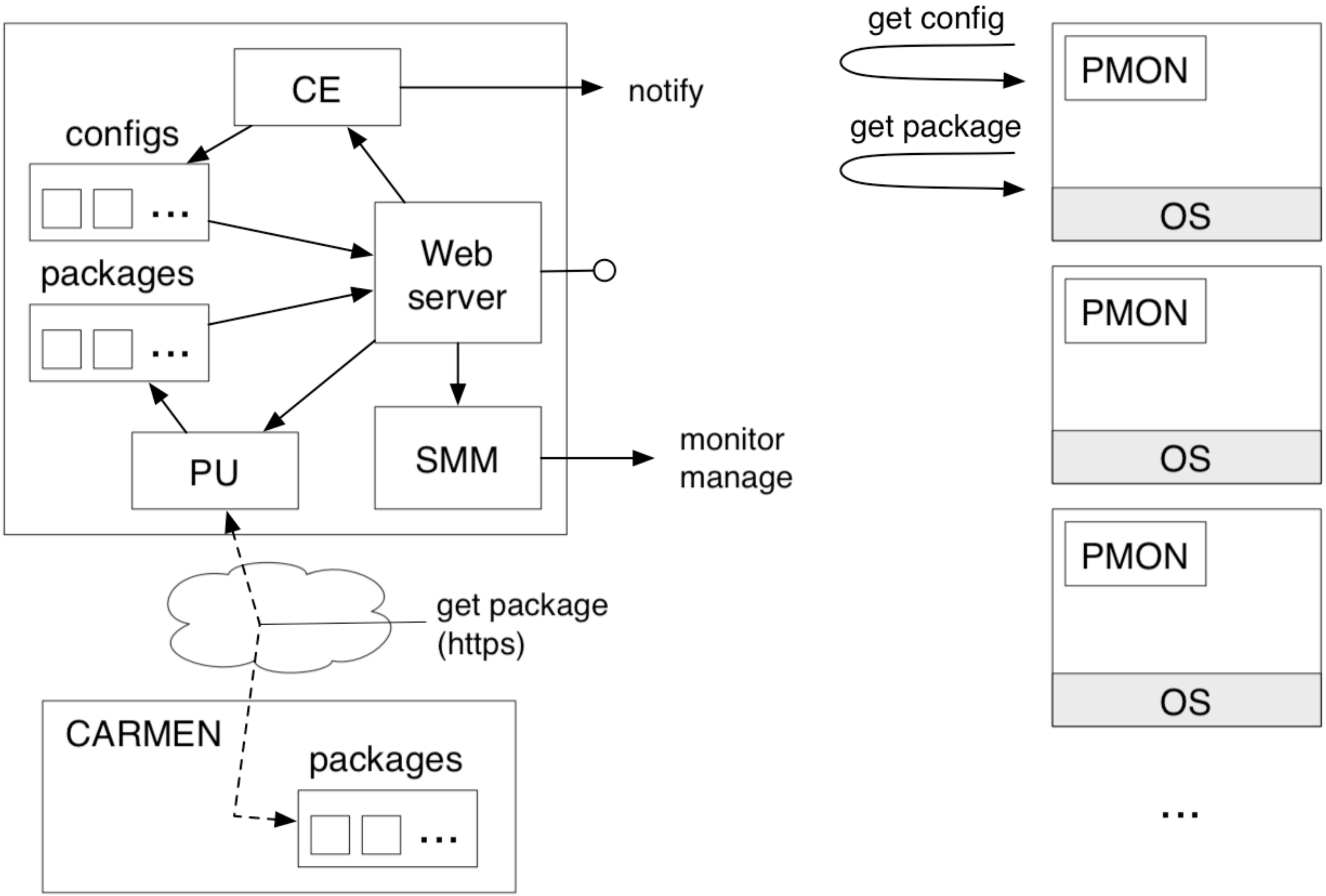
DAVE - High Availability


Applications
Applications
App srv / db client
lib
SQLxxx
Internal IP network, (redundancy)
Internal IP net
Off-site synch for disaster recovery
off-site data store (db mirror)
Db servers (clusters, redundant servers)
Sync to on-line archive
secondary data store (archive)
Storage Area Network
SAN
Disk storage (RAID) and backup systems
RAID
Backup

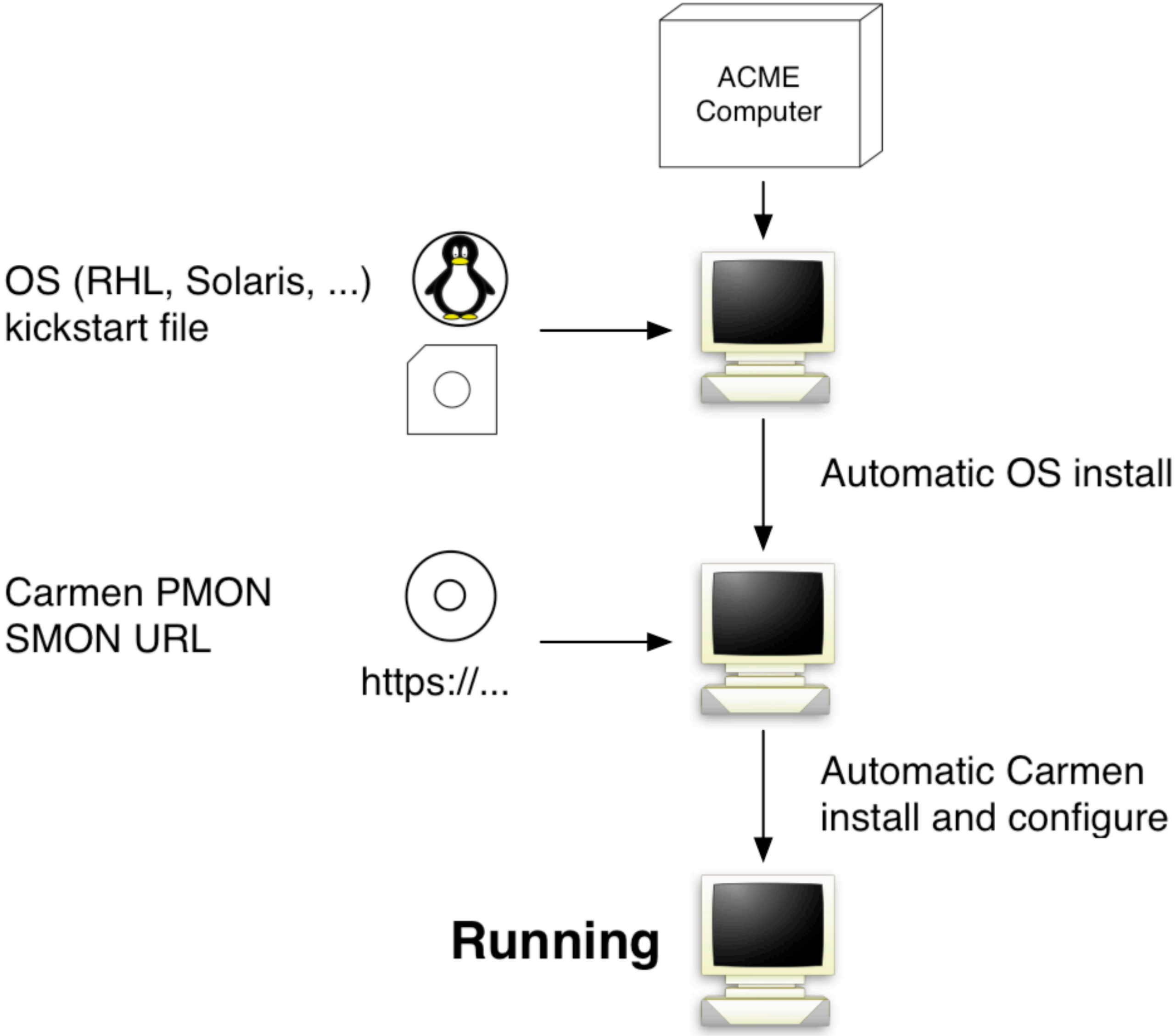
Deployment example



System services (1)



Installing a new machine



ACME Computer

OS (RHL, Solaris, ...)
kickstart file

Automatic OS install

Carmen PMON
SMON URL

https://...

Automatic Carmen
install and configure

Running

Summary

- Security = ACS + Quality
  - Much of the security design is for redundancy, load balancing and other system quality issues
  - Protection against mistakes as well as deliberate attacks
- Real security must be supported by in-depth architecture (and implementation); there just is no shortcut…
- The *network security view* is necessary as a complement to components views
  - Components and APIs for developers
  - Protocols and Data Formats are a must for system security

I conclude that there are two ways of constructing a software design: One way is to make it so simple that there are *obviously* no deficiencies and the other way is to make it so complicated that there are no *obvious* deficiencies.

C.A.R Hoare, Turing Lecture
”The Emperor’s Old Clothes”,
CACM February 1981, pp. 75-83

THE END