



PAID System Design

15-413 Software Engineering

Fall 1998

Carnegie Mellon University

Pittsburgh, PA





Overview

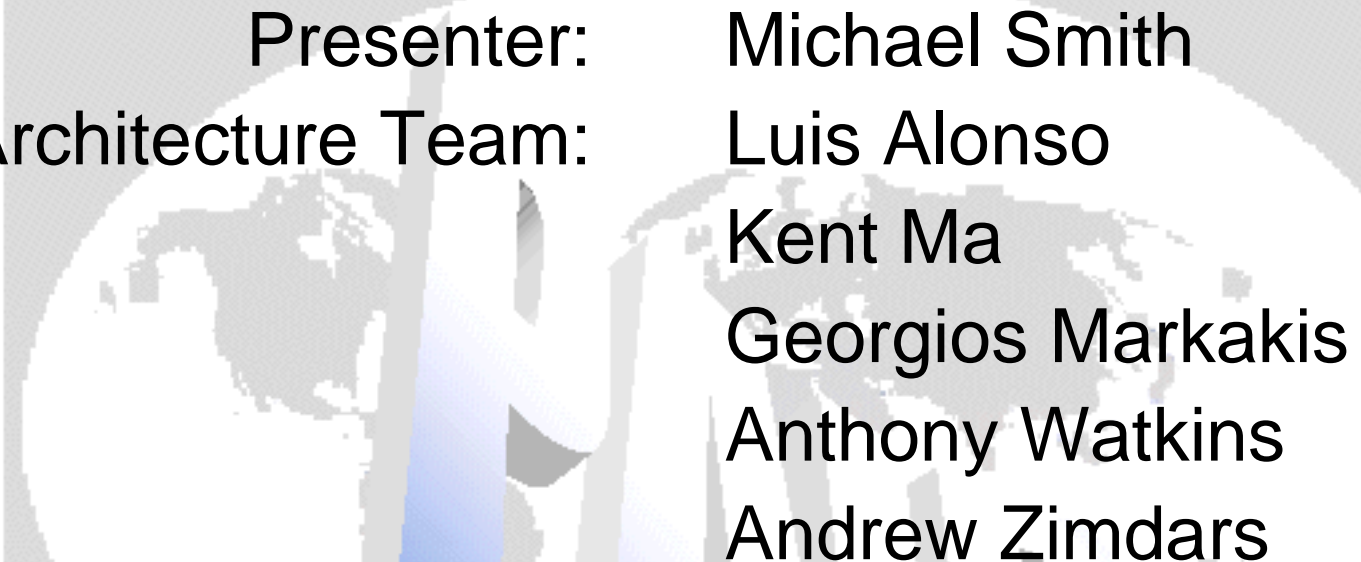
- 
- Goals and System Decomposition Michael Smith
 - Concurrency Wing Ling Leung
 - Hardware/Software Mapping Orly Canlas
 - Data Management Richard Markwart
 - Global Resource Handling Luis Alonso
 - Software Control Implementation Yun-Ching Lee
 - Boundary Conditions Euijung Ra
 - Design Rationale Ivan Tumanov



Sections 1 & 2

Goals and Tradeoffs

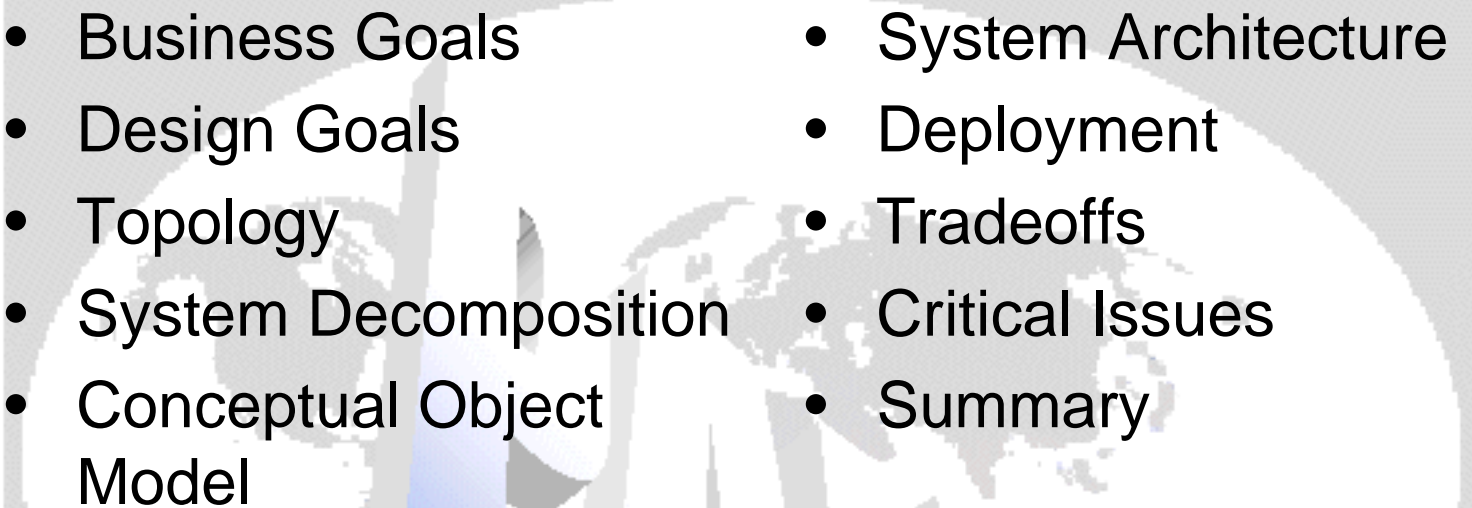
System Architecture



Presenter: Michael Smith
Architecture Team: Luis Alonso
Kent Ma
Georgios Markakis
Anthony Watkins
Andrew Zimdars



Outline

- 
- Business Goals
 - Design Goals
 - Topology
 - System Decomposition
 - Conceptual Object Model
 - System Architecture
 - Deployment
 - Tradeoffs
 - Critical Issues
 - Summary



Business Goals

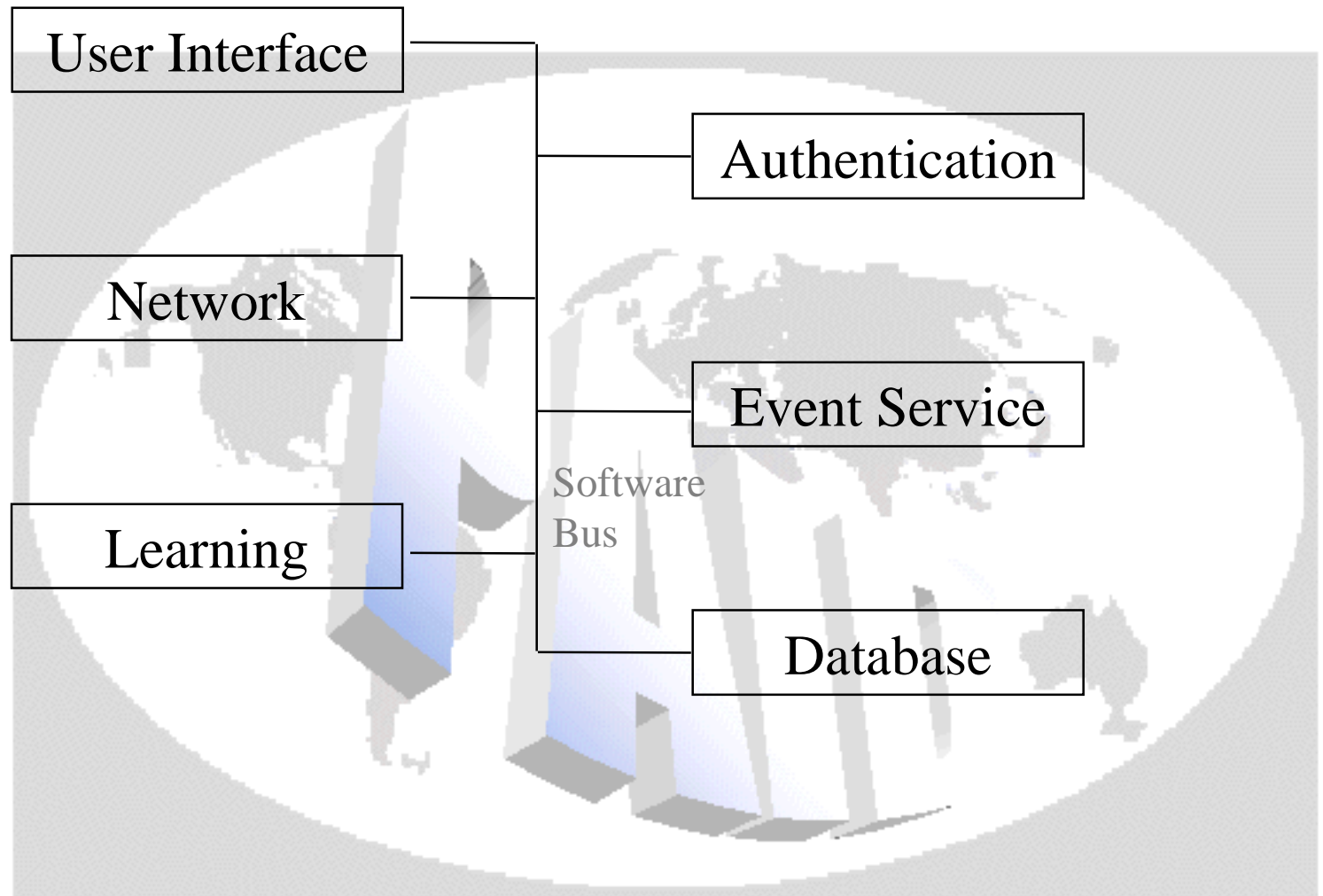
- 
- Distribution of all kinds of information
 - Low Entry & Administrative Cost
 - Easy Development of New Applications
 - Fast Response Time
 - Up To Date Information
 - Security



Design Goals

- 
- Extensibility
 - Scalability
 - Location Transparency
 - Actuality
 - Reliability
 - Adaptability

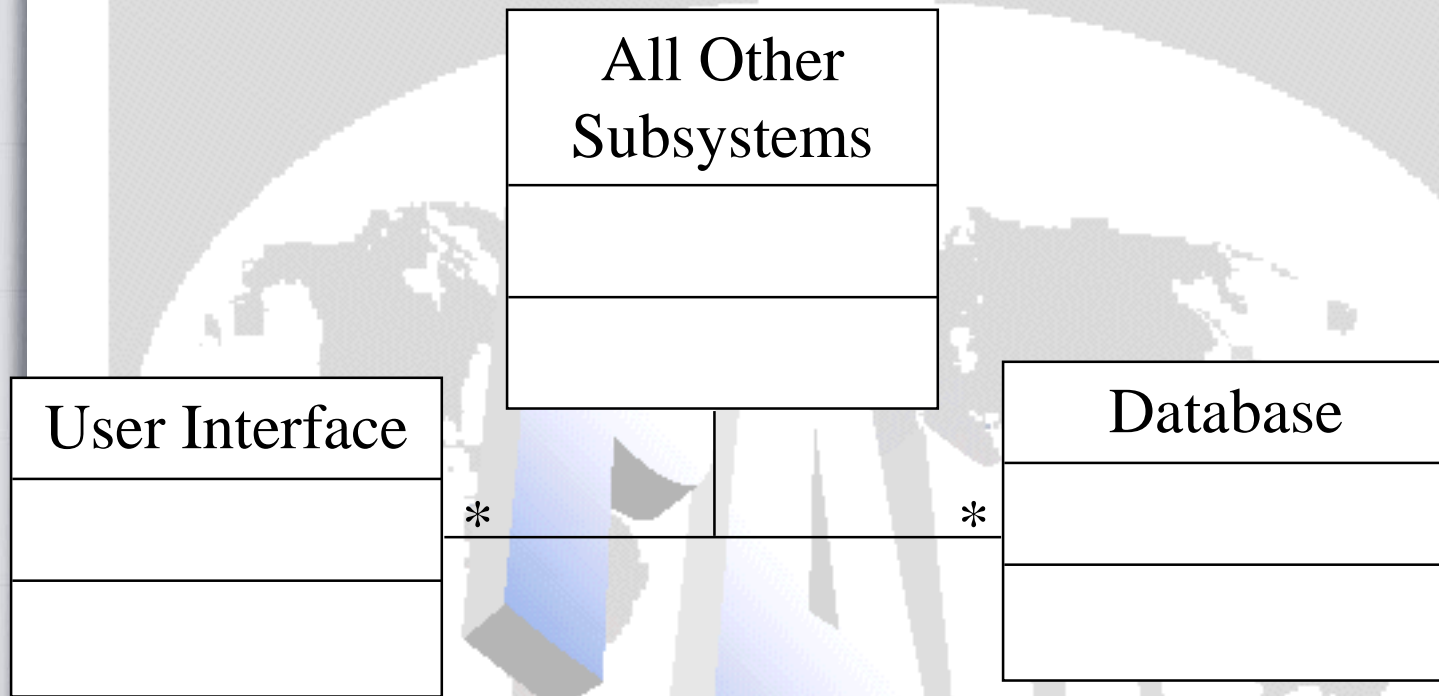
Topology



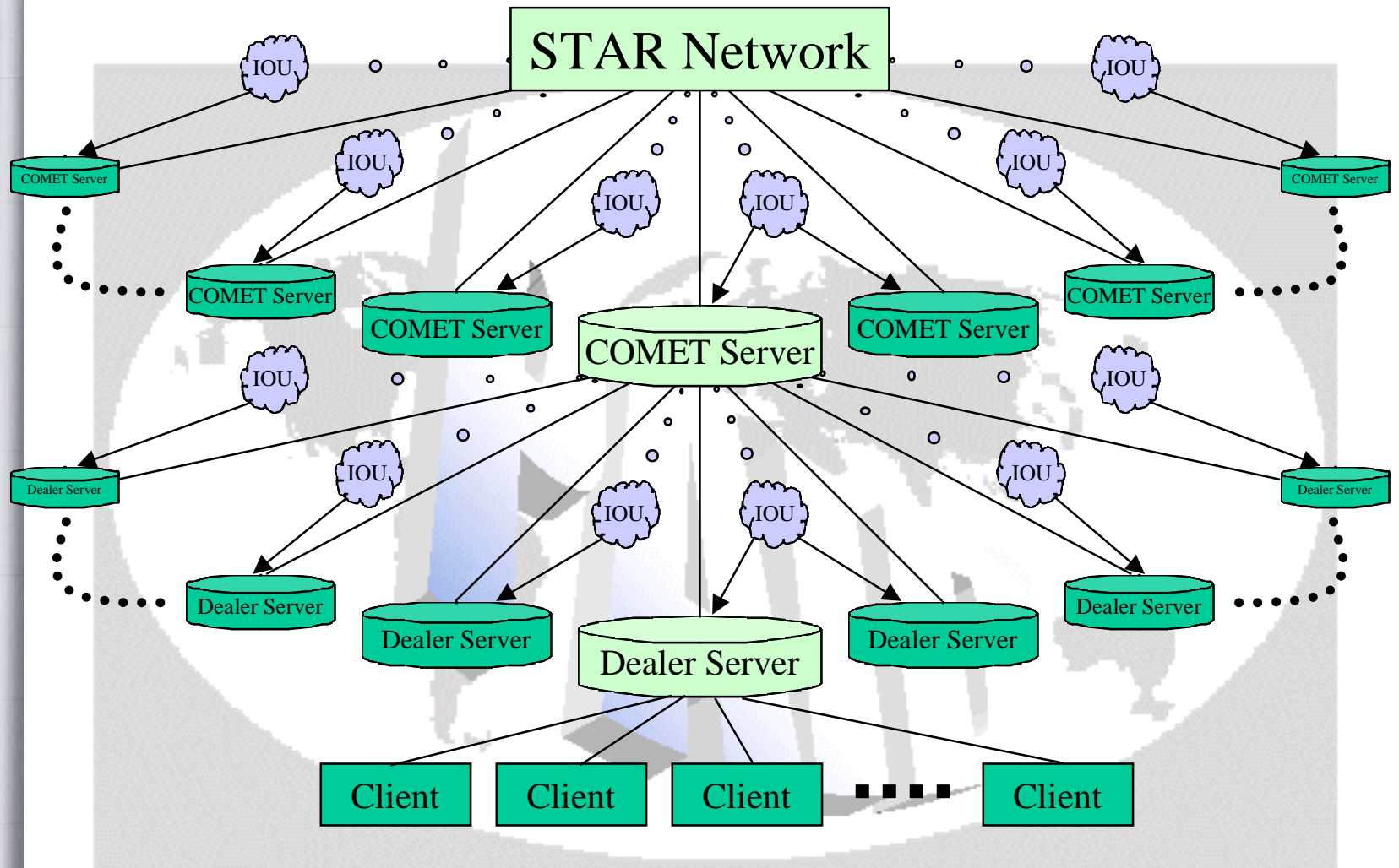
System Decomposition

- Learning
 - Network
 - User Interface
 - Database
 - Events
 - Authentication
- 
- A 3D bar chart is overlaid on a world map. The chart has six bars of varying heights, corresponding to the items in the list. The 'Authentication' bar is the tallest, colored in a gradient from blue to white, indicating a high value or priority. The other bars are shorter and colored in shades of gray. The world map is rendered in a light gray tone, showing the continents.

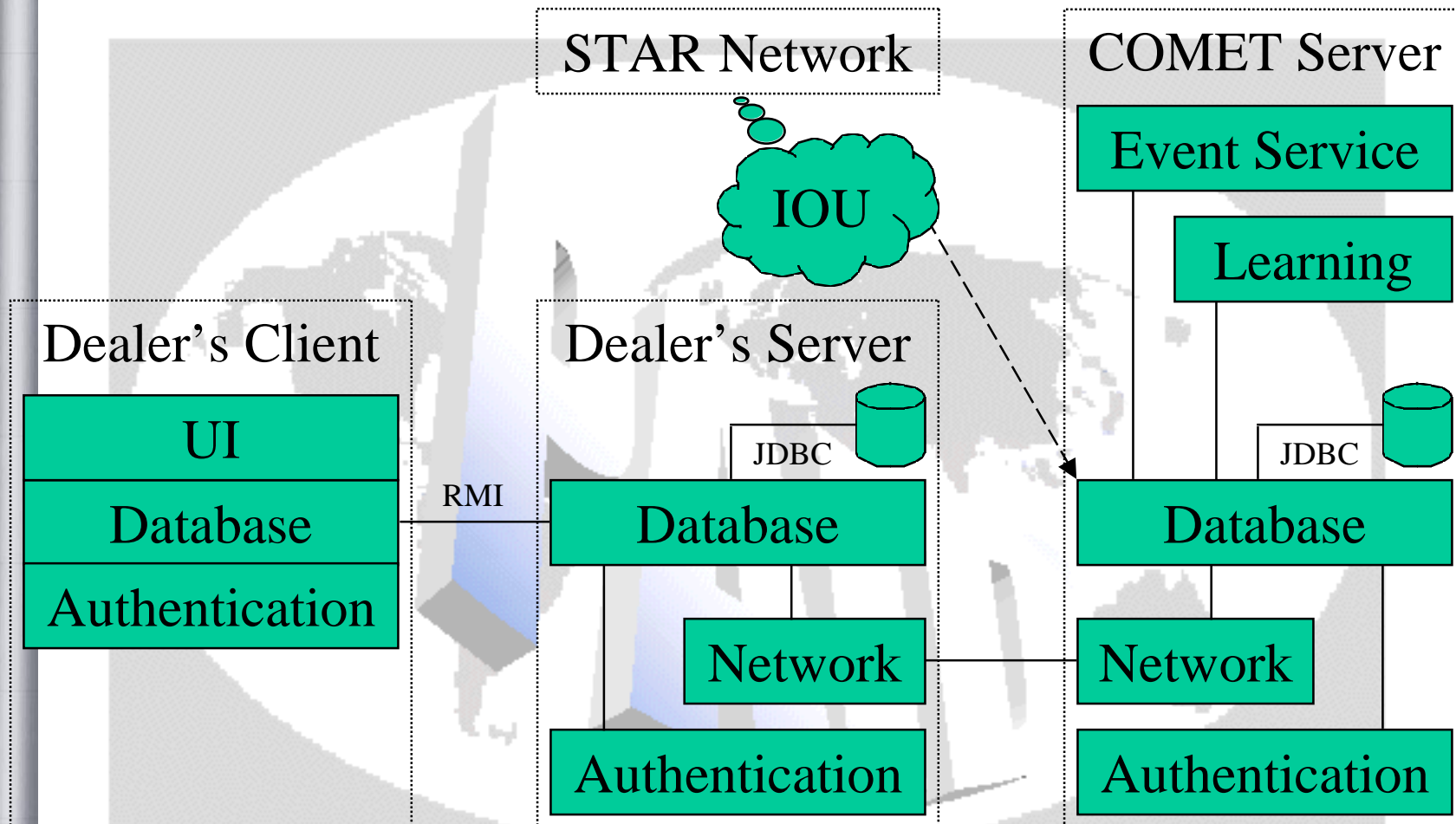
Conceptual Object Model



System Architecture

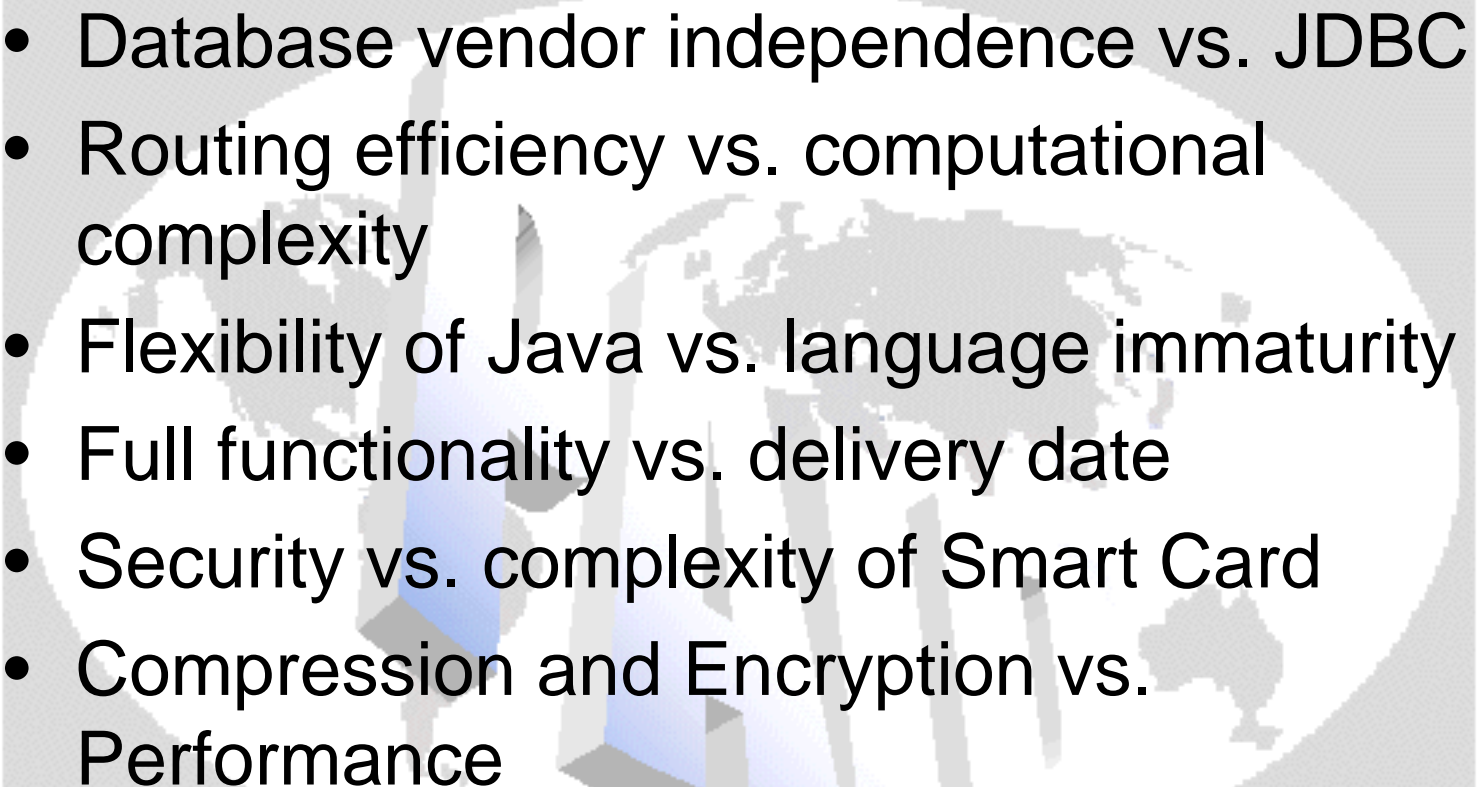


Deployment



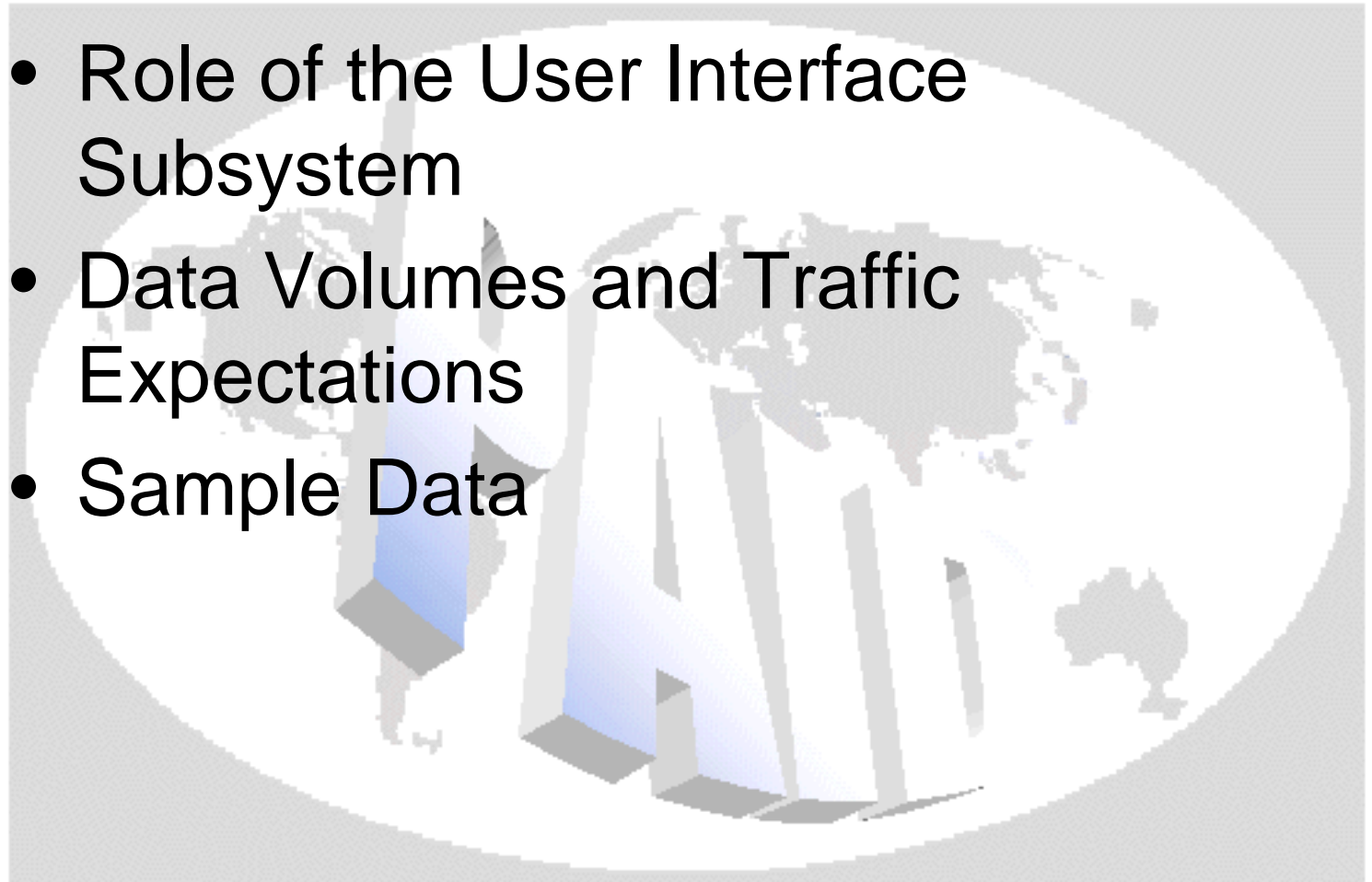


Tradeoffs

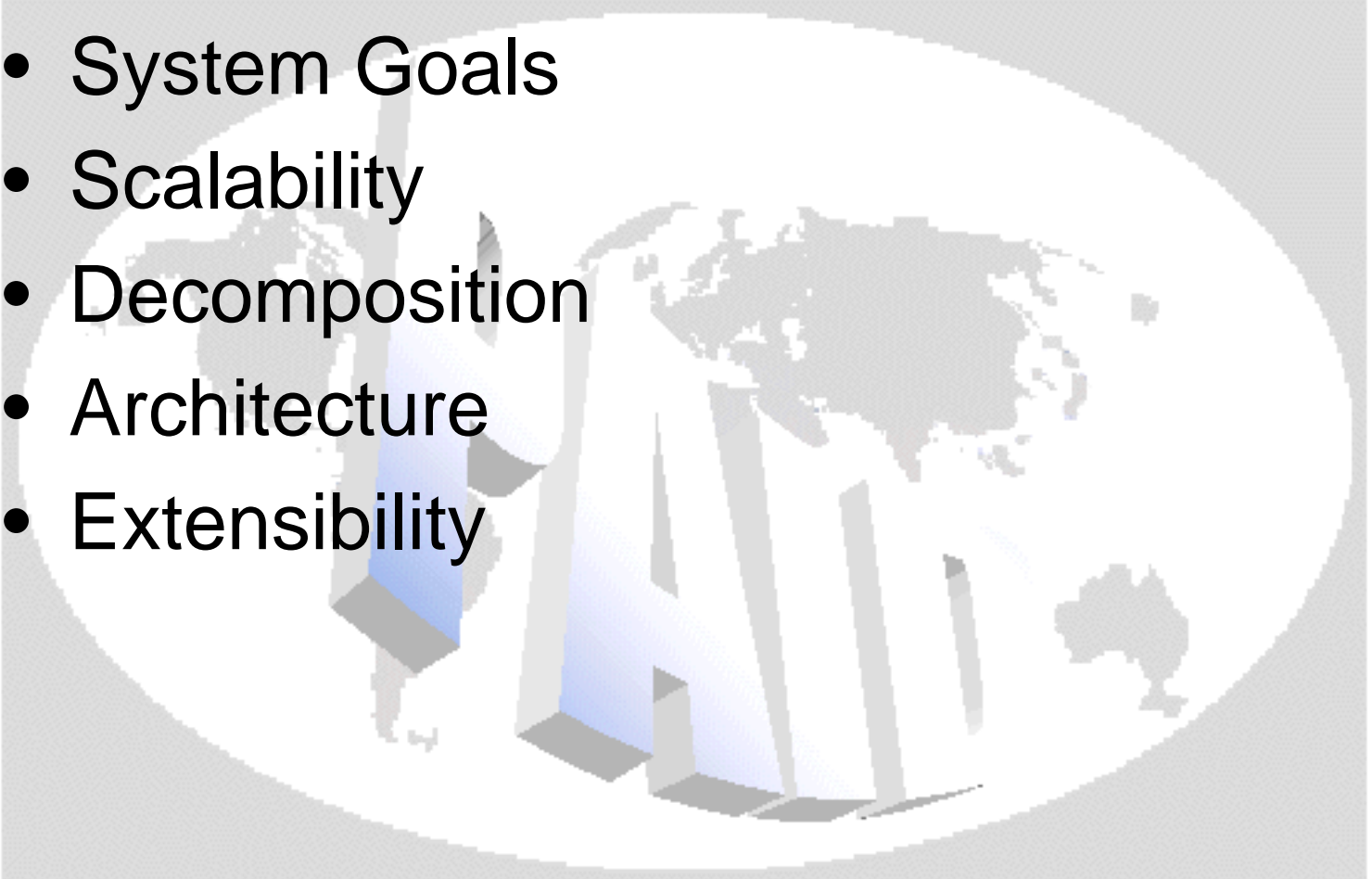
- 
- Database vendor independence vs. JDBC
 - Routing efficiency vs. computational complexity
 - Flexibility of Java vs. language immaturity
 - Full functionality vs. delivery date
 - Security vs. complexity of Smart Card
 - Compression and Encryption vs. Performance

Critical Issues

- Role of the User Interface Subsystem
- Data Volumes and Traffic Expectations
- Sample Data



Summary

- System Goals
 - Scalability
 - Decomposition
 - Architecture
 - Extensibility
- 



Section 3. Concurrency

Identification

Teams: Learning and Event Service

Presenter: Wing Ling Leung

Members: Jonathan Hsieh

James Lampe

Yun-Ching Lee

Rudy Setiawan

Jonathan Wildstrom

Andrew Zimdars





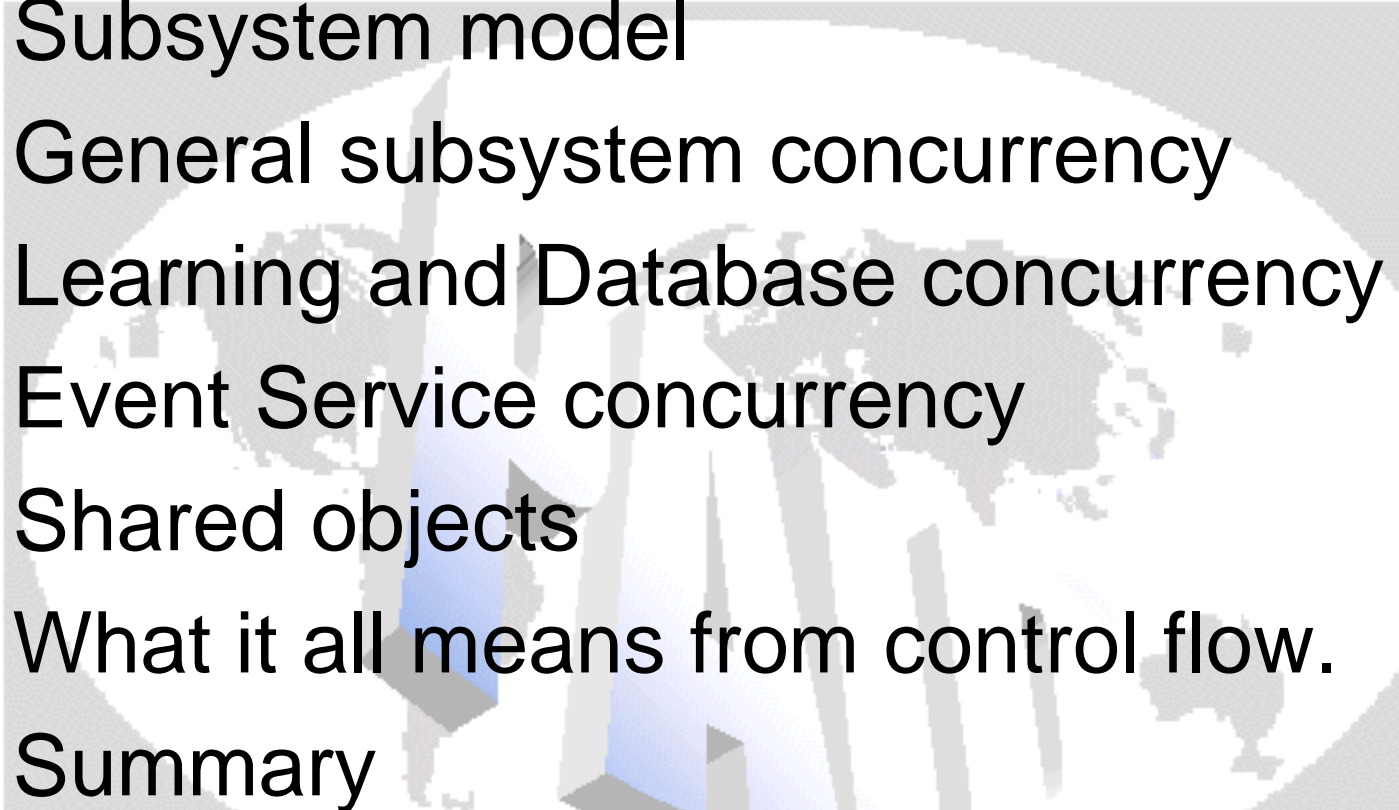
Learning Team Members:

- 
- Jonathan Hsieh
 - James Lampe
 - Yun-Ching Lee
 - Wing Ling Leung
 - Rudy Setiawan
 - Jonathan Wildstrom
 - Andrew Zimdars

 - Eric Stein - Learning Team Coach

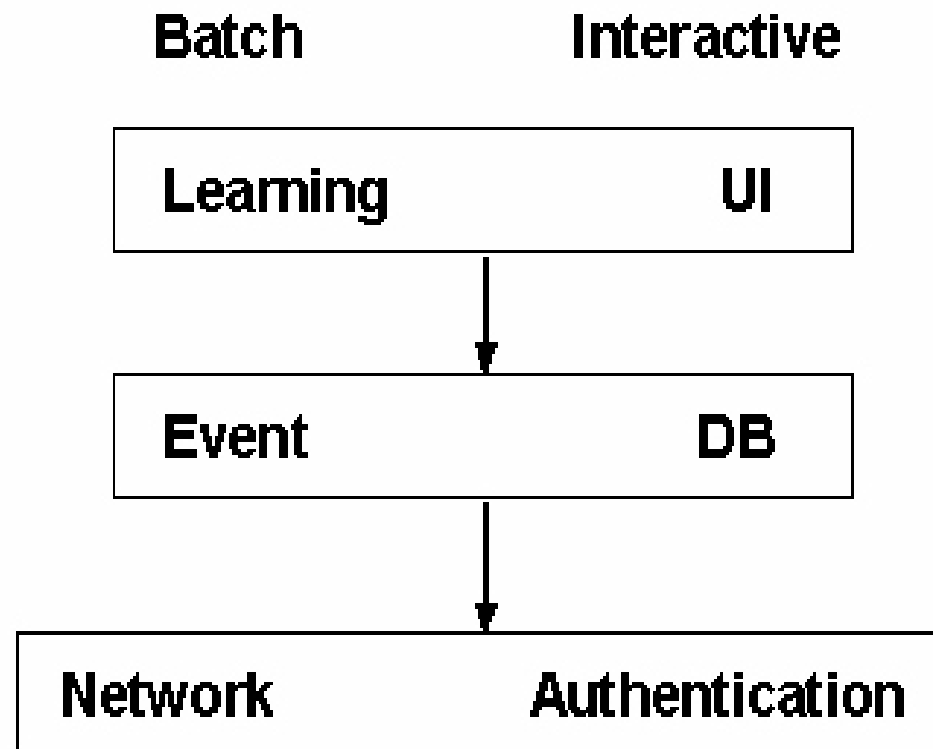


Outline

- Subsystem model
 - General subsystem concurrency
 - Learning and Database concurrency
 - Event Service concurrency
 - Shared objects
 - What it all means from control flow.
 - Summary
- 

Layer Model

Layer Model





Concurrency Identification Overview

- UI initially triggers certain events from user input, then control is passed procedurally from subsystem to subsystem within PAID.
- The nature of the way PAID subsystems interact as a whole eliminates possible concurrency problems.
- All PAID subsystems can run concurrently.



Concurrency Identification

Learning and Database

- Both Learning and Database will run as individual processes outside of the regular flow of control passing. However, since both do not share objects nor interfere with other subsystems, neither subsystem introduces possible concurrency issues.



Concurrency Identification

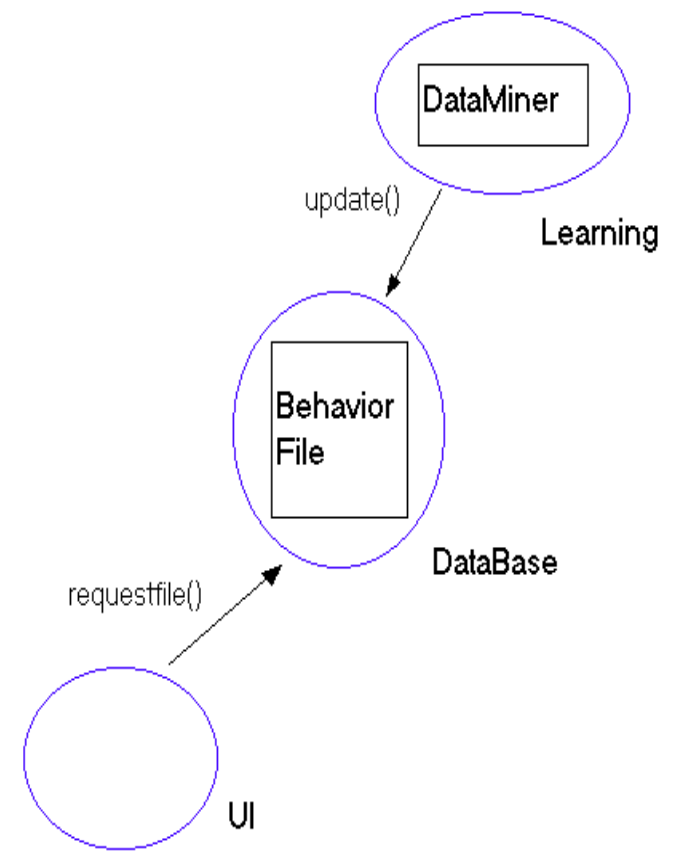
Event Service

- Event Service only acts as the channel for communication among the subsystems.
- Event Service does not interfere with the workings of other subsystems and therefore runs concurrently with all other subsystems.
- Although UI may broadcast events of user actions, users have no direct interaction with Event Service or any other subsystem besides UI or Authentication.

Concurrency Identification

Shared Object Issues

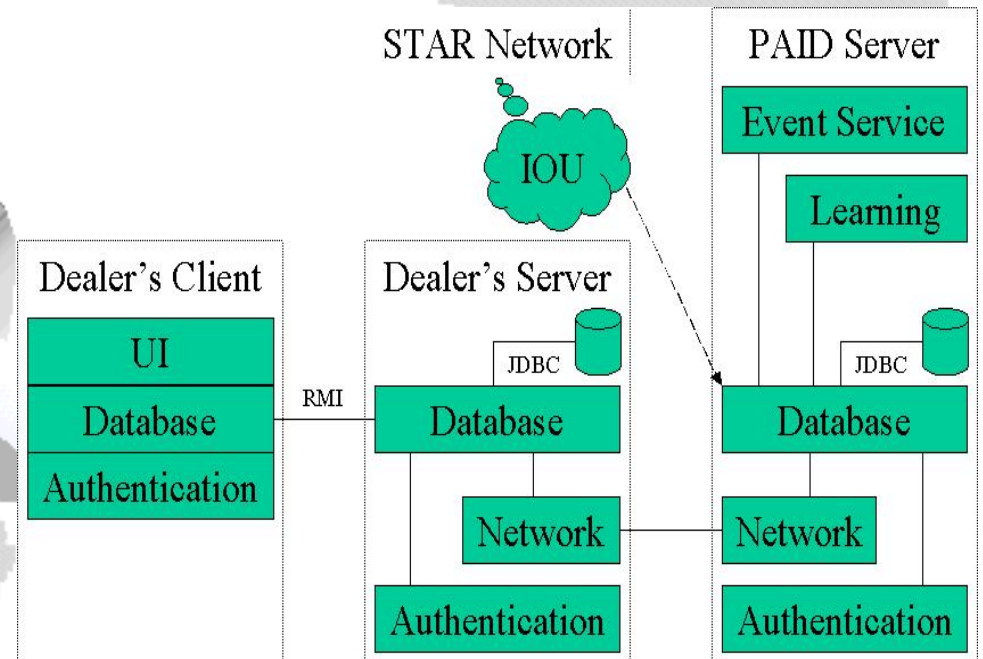
- One shared object in the current design is the BehaviorFile. It is updated by Learning and provides other subsystems with intelligent recommendations. Database locking mechanisms will handle all such concurrent accesses correctly.



Concurrency Identification

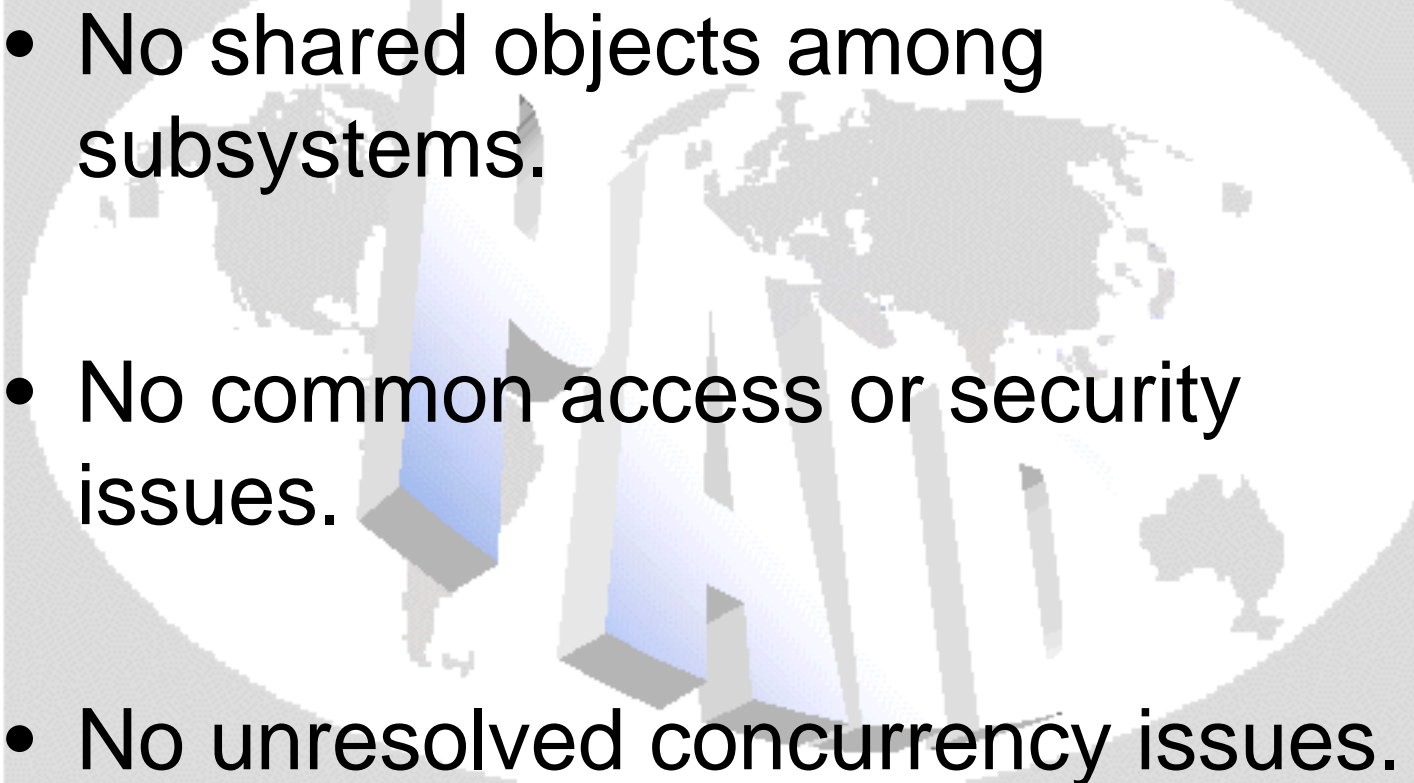
What it all means

- Control is passed procedurally for all user actions in the PAID system.
- Subsystems do not interfere with workings of other subsystems.



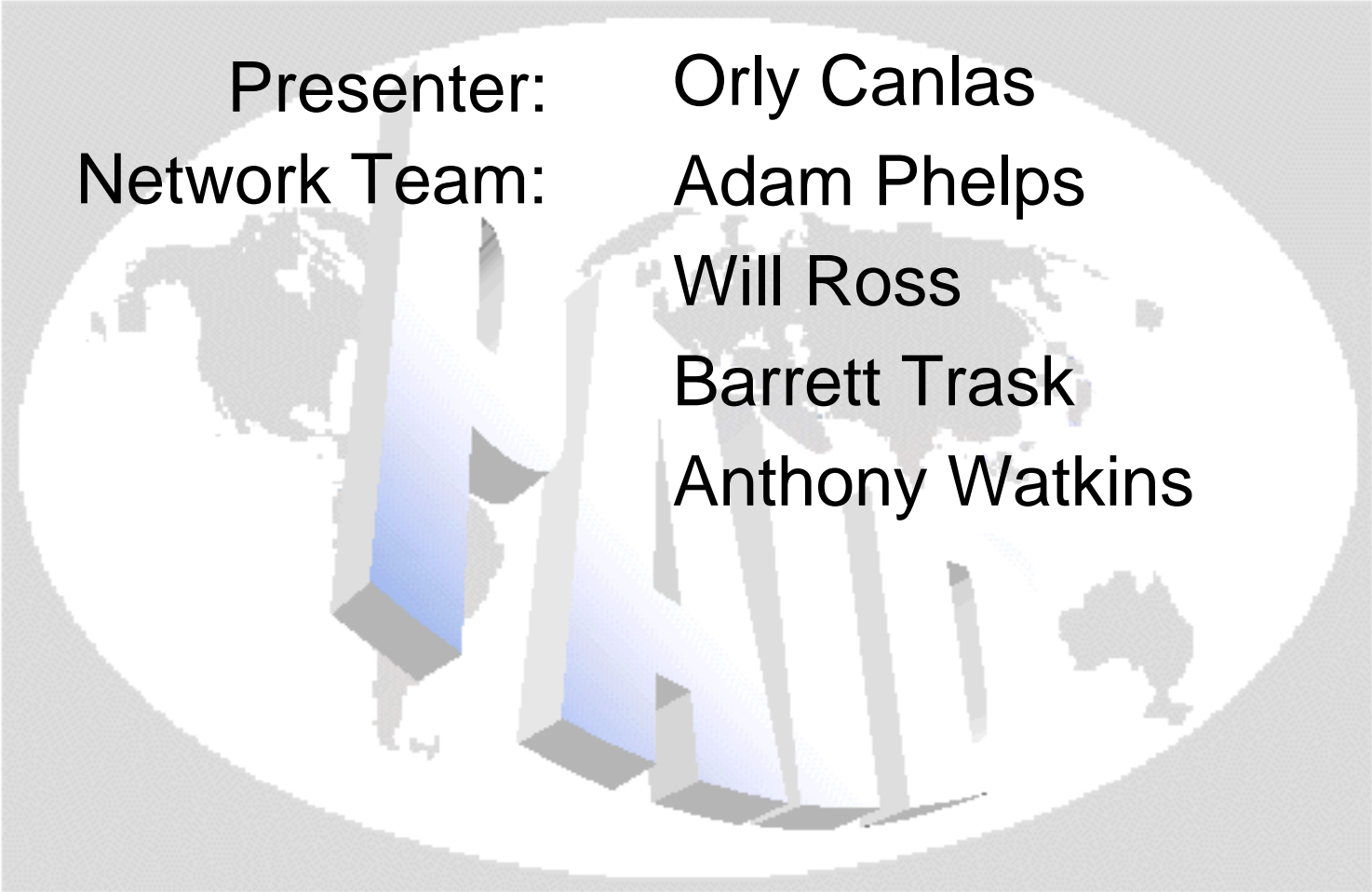


Concurrency Identification Summary

- 
- No shared objects among subsystems.
 - No common access or security issues.
 - No unresolved concurrency issues.

Section 4

Hardware/Software Mapping

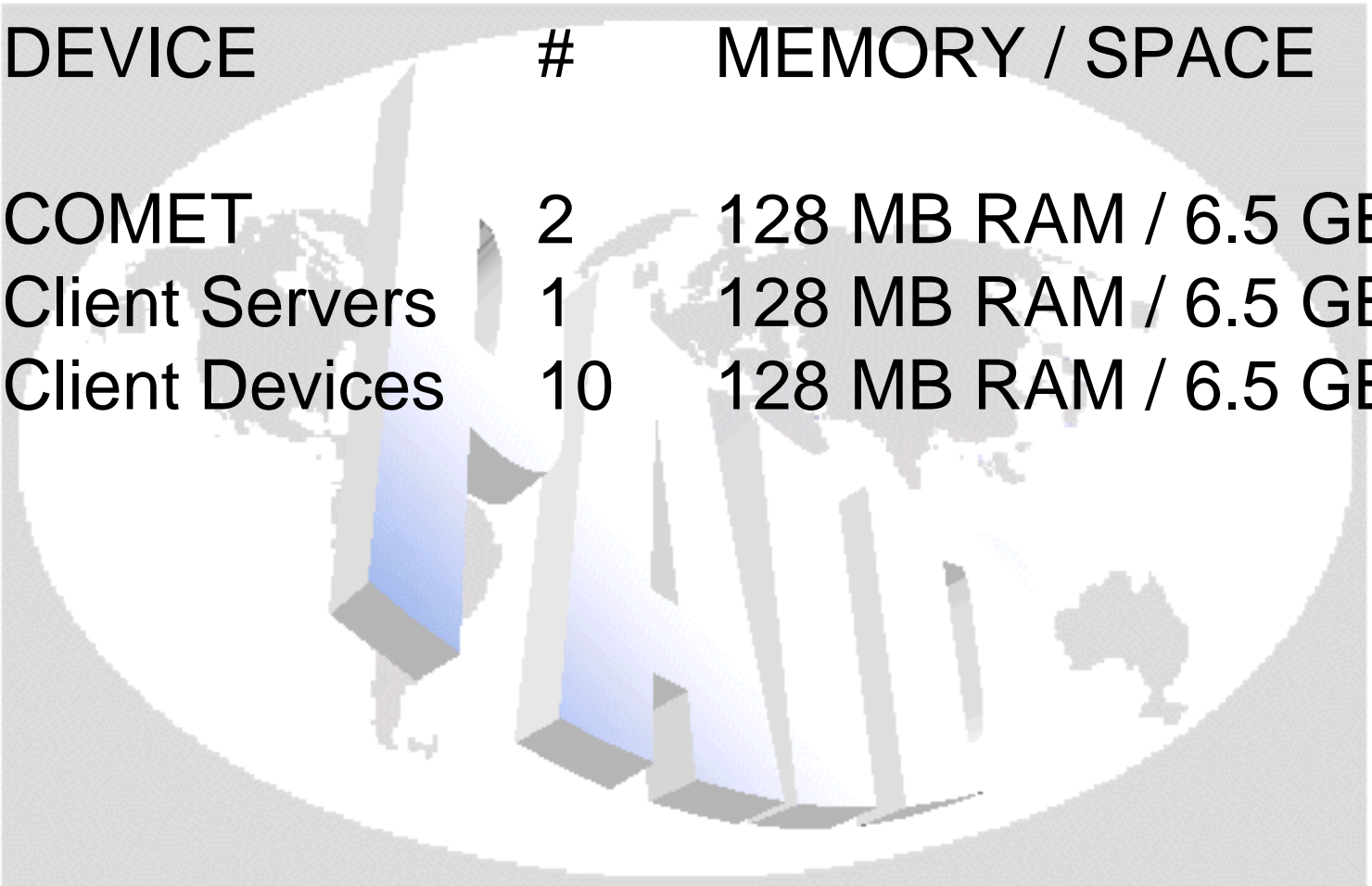


Presenter: Orly Canlas
Network Team: Adam Phelps
Will Ross
Barrett Trask
Anthony Watkins

Prototype Specifications

- Development Tool : Voyager 2.0
- Development Platforms : Windows NT & Linux
- PAID (COMET) Server specification
 - Pentium-II 400 MHz with 128 MB RAM
 - No additional hardware for compression and encryption
 - SMARTCard Readers for login
- Generally available PDA technology is on the threshold of supporting PAID
- Wireless communication will not be prototyped this semester

Prototype Devices

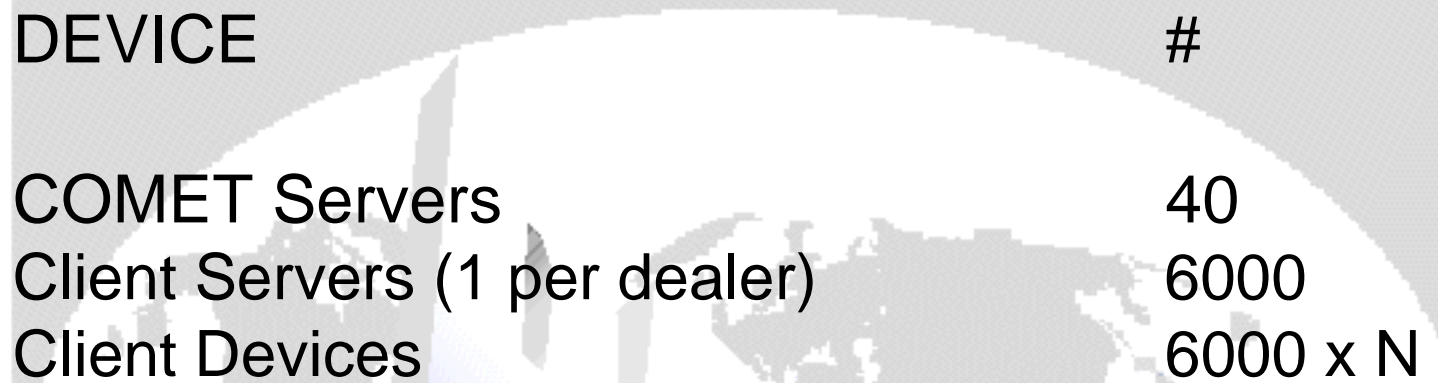


DEVICE	#	MEMORY / SPACE
COMET	2	128 MB RAM / 6.5 GB
Client Servers	1	128 MB RAM / 6.5 GB
Client Devices	10	128 MB RAM / 6.5 GB

Final System Specifications

- Response Time
 - 90% of local requests should take less than 10 seconds
 - Aiming for a maximum response time of 1 minute, not including time taken to connect to the network, for network requests
- Additional hardware which may be necessary for data mining performed by the Learning subsystem

Final System Devices



DEVICE	#
COMET Servers	40
Client Servers (1 per dealer)	6000
Client Devices	6000 x N

Typical Updates for FDOK

vehicle	area	Updates Size	DB Size	#vehicles
utility	Europe w/o	9.8 MB	43 MB	23000
utility	Germany	9.4 MB	34 MB	15000
utility	others	0.8 MB	2.1 MB	2000
passenger	Europe w/o	2.2 MB	14.5 MB	35000
passenger	Germany	3.8 MB	20 MB	48000
passenger	America (N+S)	1.2 MB	10 MB	21000
passenger	others	0.2 MB	0.9 MB	2000

Typical Updates for EPC

FILE TYPE

Image Files

Database Updates

DATA GENERATION

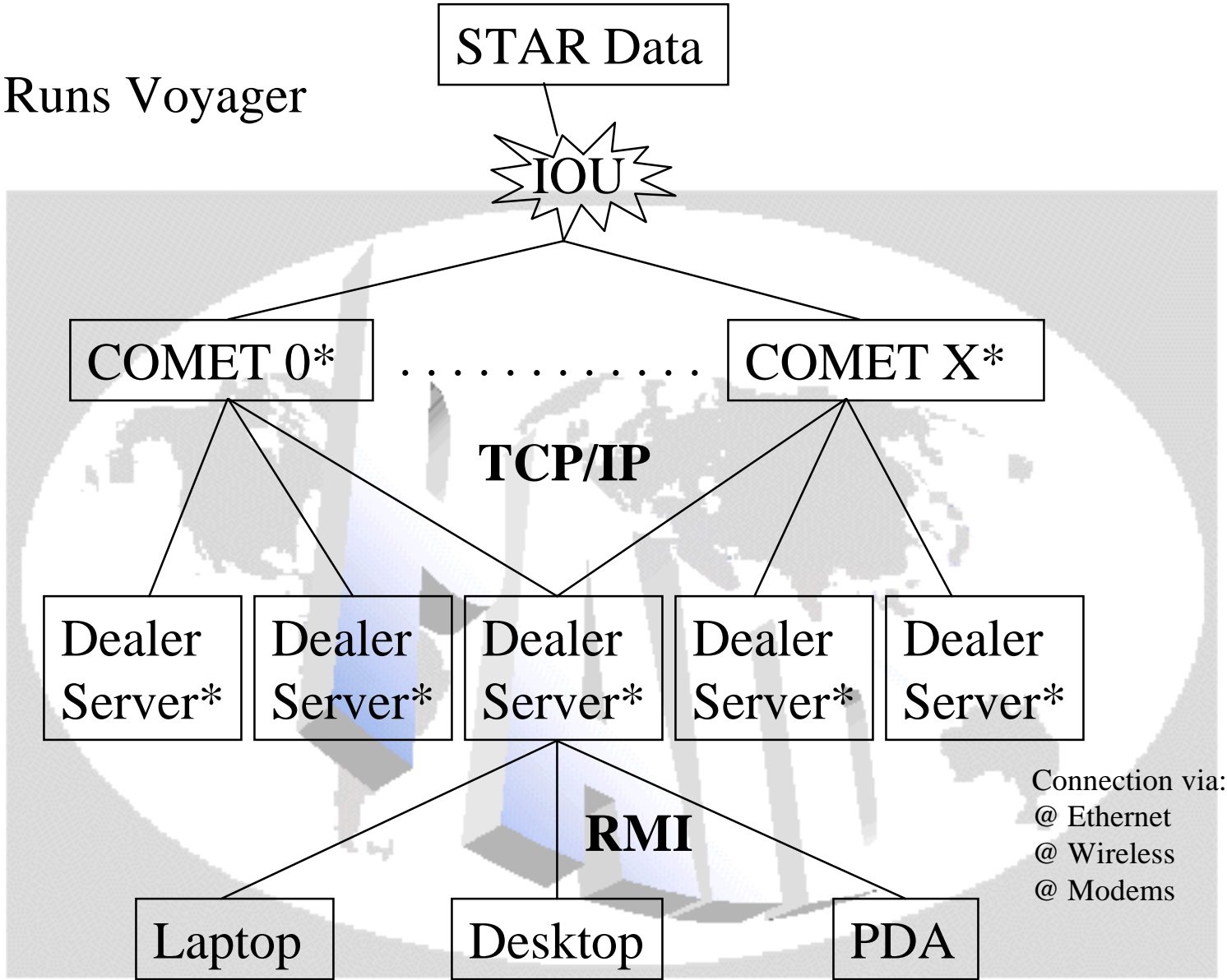
ca. 7MB / week.

ca. 3MB / week.



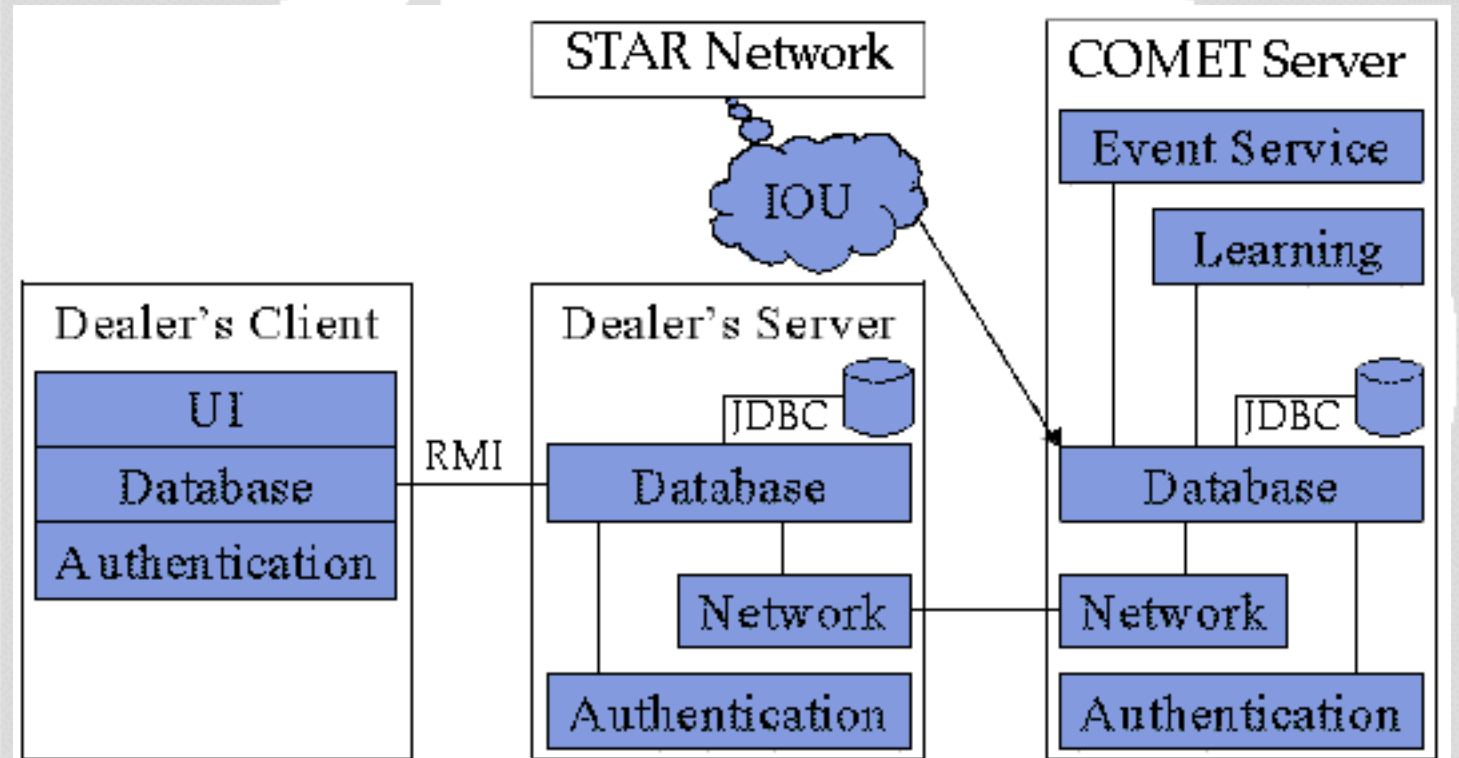
Connectivity

* Runs Voyager



Connectivity

Unlabeled connections are made using Voyager

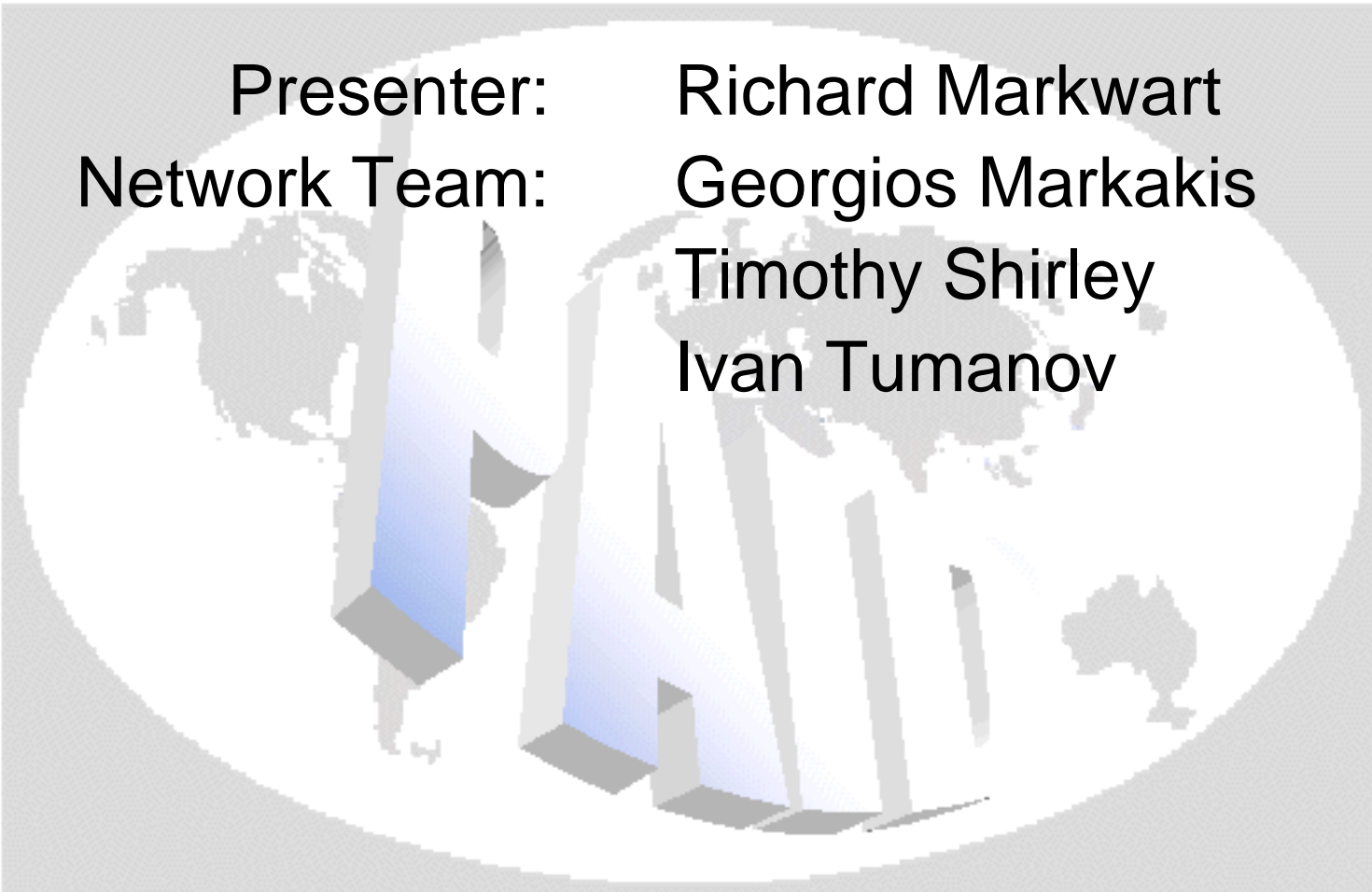


Voyager

- 100% Java distributed computing platform
- Allows any Java class to be remotely enabled
- Integrated native CORBA support
 - Can communicate with other CORBA systems regardless of implementation language
- Messaging features, including parallel multicast and publish-subscribe
- Multi-layered, scalable architecture
- Mobility
 - Locality Optimization
 - Disconnected mode
- Supports standard Java security manager system
- Simple and robust (supported by LINUX)

Section 5

Data Management



Presenter: Richard Markwart
Network Team: Georgios Markakis
Timothy Shirley
Ivan Tumanov

Outline

- Goals
 - Solutions
 - Organization of Data
 - Error Handling
 - Local Data Management
 - Future Concerns
- 

Goals

- **Data integrity is the paramount goal**
 - We need to ensure accurate data
 - Manual recovery of data is extremely costly, so loss of data *must* be prevented
- **Location Transparency**
 - Users/Other systems do not have to know where data is stored to access it
- **Extensibility**
 - Easy adaptation to new requirements
 - Ability to leverage new technologies
 - Extensibility requires that the system not be tied to a particular vendor or platform

Solutions - Extensibility

- Database is viewed externally as an object database accessed through an API provided by the Data Management package
 - Other subsystems need not concern themselves with the structure of the database, the database vendor (Oracle, Sybase, etc.), or even the type of database (relational, object, etc.) used
 - JDBC used internally to hide the details of databases
 - Allows the use of any database that supports JDBC
 - Allows us to change databases easily
 - One database can be used for development and prototyping and a different one for the actual production system
 - This limits us to those database features directly supported by JDBC
 - Any features not supported by JDBC will be implemented directly by the Data Management system

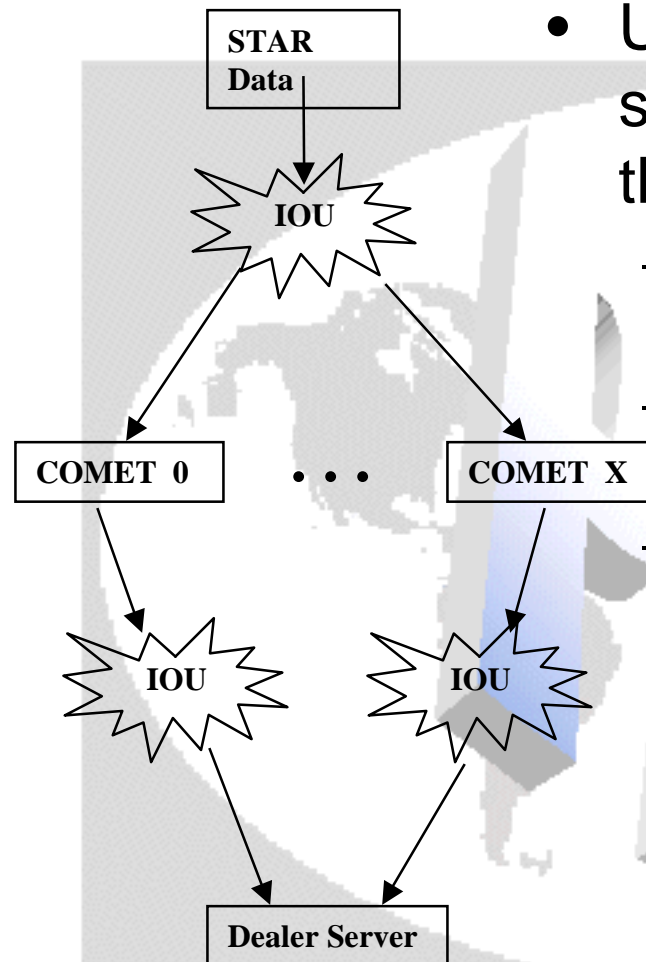
Solutions - Data Integrity

- Use of a proven database which supports JDBC
 - Interbase v. 5 to be used in prototype
- Data Management system will ensure that proper relationships within database are maintained
- Recovery procedures to ensure that any data loss can be repaired quickly
 - These are covered in Section 8: Boundary Conditions

Organization of Data

- Data divided into subsets
 - Examples of possible subsets:
 - Classes of vehicles (e.g. M-class)
 - Location where vehicles were sold (e.g. All vehicles sold in Germany)
 - Date of sale (e.g. all vehicles sold in 1997)
 - Each data request requires one or more subsets
 - The subsets required to satisfy a request will be determined by the request itself
 - This means that the data management system will know what parts of a request can be filled locally and what parts must be obtained from a COMET server before the database is accessed.
 - Each COMET and dealer server will store one or more subsets locally
 - Every subset of data will be stored on at least one COMET server

Organization of Data



- Updates to a given subset are only sent to those servers which store that subset
 - Updates sent to COMET servers via IOU
 - COMET servers forward the IOU updates to the dealer servers
 - If a dealer server is offline when an update is sent, then that update is queued by the COMET server, and is sent when the dealer server goes online
 - Only a limited number of updates can be queued, so if a dealer server is offline for too long it will miss updates

Organization of Data

- Each dealer server will register with a particular COMET server to receive updates for a particular subset
 - That COMET server is responsible for sending updates to that subset to the dealer server
 - This is accomplished by simply forwarding the IOU's that the COMET server receives from STAR Data
 - A dealer server might register with different COMET servers for different subsets
 - Example: Klaus's dealer server could receive data on all vehicles sold in 1996 from "COMET-1", and data on all vehicles sold in 1997 from "COMET-2"
 - This registration process takes place in the background, requiring no interaction from the user



Data Request from COMET Server

- If a data request requires data not stored locally, the missing data must be obtained from a COMET server
- If the requested data is cached on the local machine, then the cached copy is used
- If not, then the request is encoded and passed to the network subsystem for transmission to a COMET server
 - Network is given a list of COMET servers which can fill the request
 - Network selects a server and sends the request to the server
 - The server carries out the request, and returns the result to the dealer server
 - The results are cached for future use
 - If no single COMET server can fill the request, then the data management system will divide the request into multiple requests, each of which can be filled on a single COMET server

Error Handling

- Sources of Error:
 - Physical damage to server (e.g. accident in dealership, electrical surge, etc.)
 - Failure to obtain updates for long period of time
 - A limited number of updates can be queued, but eventually the queue will overflow, and then updates will be lost
 - Possible Solutions:
 - Reinstall data set
 - This will essentially be the same as reinstalling the system
 - This is the required solution in the case of physical damage and similar errors
 - Dedicated IOU Server. This server's role would be to simply store all IOU's generated by STAR Data. This server could then be contacted to obtain any number of missing updates



Local Data Management

- Data Management system is also responsible for exposing the local data storage system (e.g. the local file system) to other PAID systems
- Current design includes an API for generic data storage
 - More specific methods will be created based on the needs of other subsystems as those needs become apparent
 - Because of the ad hoc nature of these methods, only methods which can be implemented quickly, reliably, and easily will be allowed
 - The data management system will ensure the platform independence of any data storage method, and will disallow methods that cannot be implemented in a platform-independent way



Future Concerns

- Object Databases
 - Emerging technology, but on the threshold of being robust enough for current use in large volume applications (such as PAID)
 - Might eventually offer significant benefits over using the relational model exposed by JDBC
- Leveraging Future Capabilities of Smart Cards/PDA's
 - Expanded capacity of these devices may allow shifting of of some permanent data onto them

Database Size Info

- **FDOK Data Size**

- Updates are a total over 5 weeks
- All size figures are in MB

Vehicle	Area	Update Size	Total Size	# Vehicles
Utility	Europe w/o	9.8	43	23000
Utility	Germany	9.4	31	15000
Utility	Others	0.8	2.1	2000
Passenger	Europe w/o	2.2	14.5	35000
Passenger	Germany	3.8	20	48000
Passenger	America (n/s)	1.2	10	21000
Passenger	Others	0.2	0.9	2000

- **EPC Data Size**

Worst Case Scenario:

Receiving All Updates

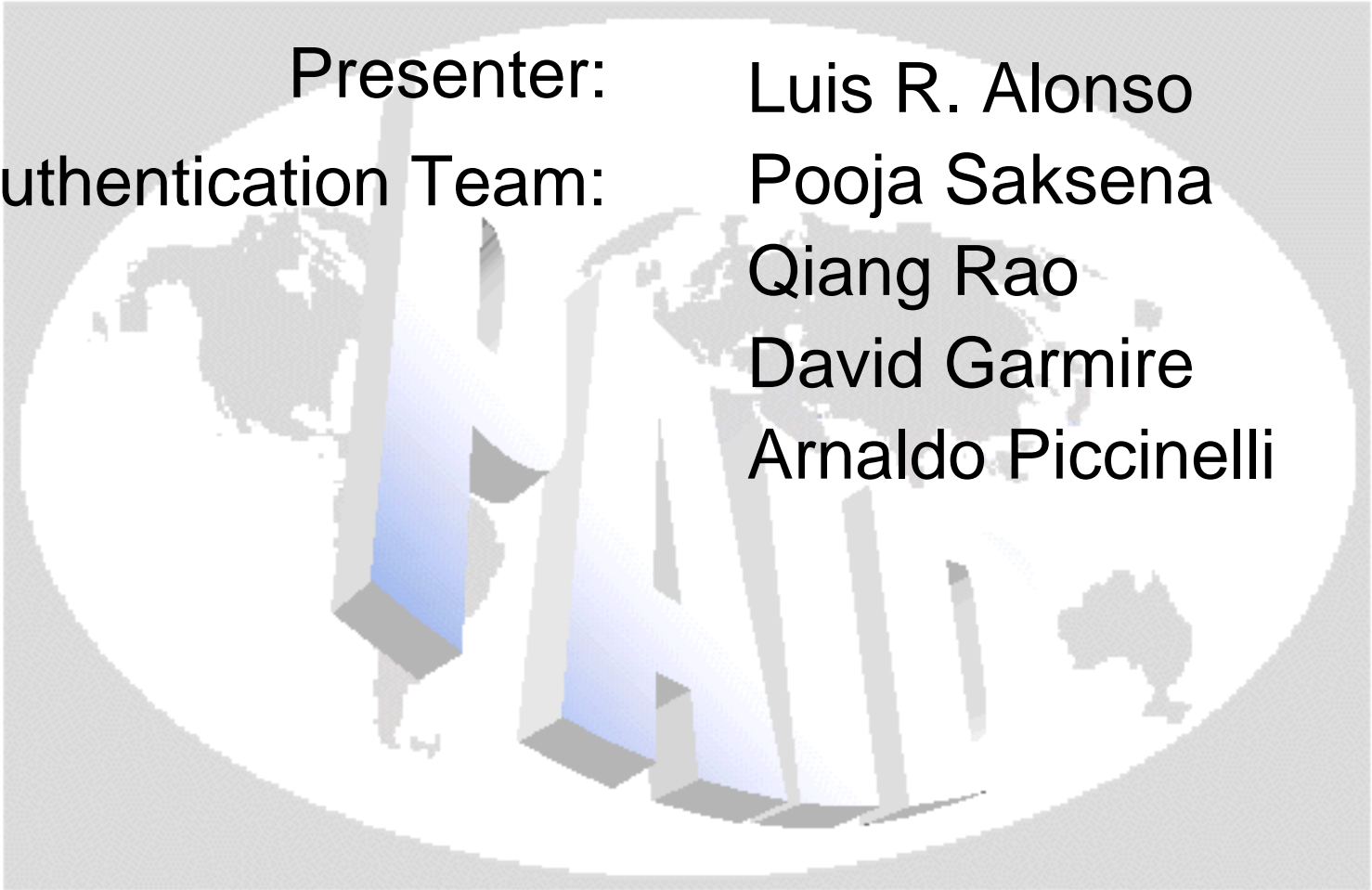
Database Updates: 3 MB/week

Image Updates: 7 MB/week



Section 6

Global Resource Handling

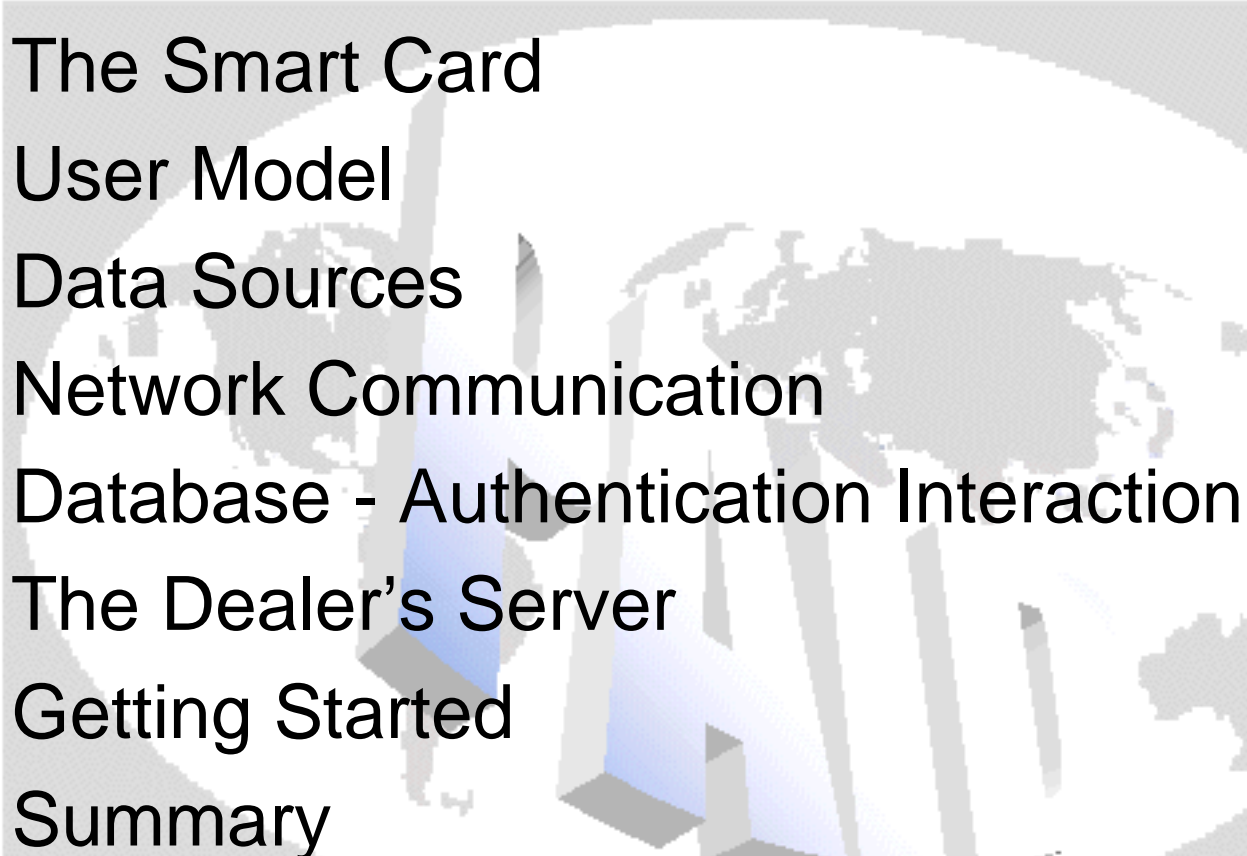


Presenter: Luis R. Alonso

Authentication Team: Pooja Saksena
Qiang Rao
David Garmire
Arnaldo Piccinelli



Outline

- Goals
 - The Smart Card
 - User Model
 - Data Sources
 - Network Communication
 - Database - Authentication Interaction
 - The Dealer's Server
 - Getting Started
 - Summary
- 



Goals

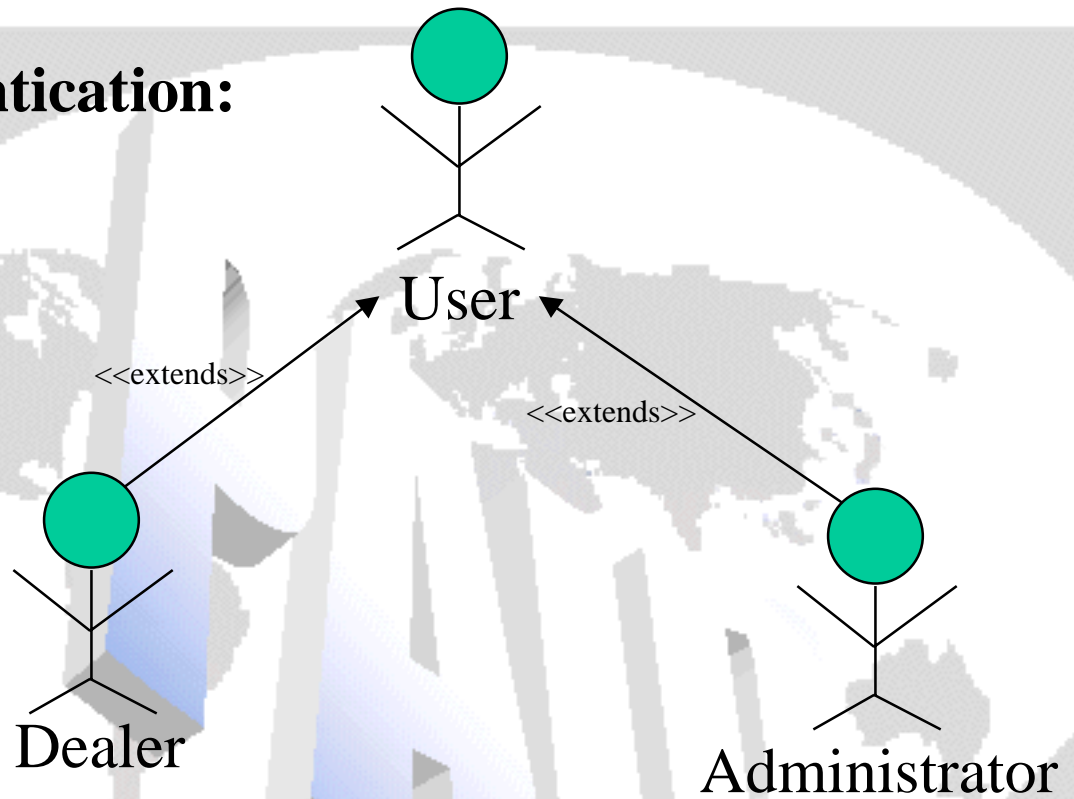
- 
- To ensure authorized access to the material stored in Daimler-Benz databases.
 - To protect the transmissions of sensitive data over public networks.

User Model

- Authentication subsystem recognizes two main groups of users
 - Administrators - rights to access other user accounts
 - Daimler-Benz manager
 - System administrators
 - Dealers - rights to access specific data in the system
 - Affiliated dealers
 - Non-affiliated dealers
 - Mechanics
 - Dealership employees
- No access for non-Daimler Benz authorized users

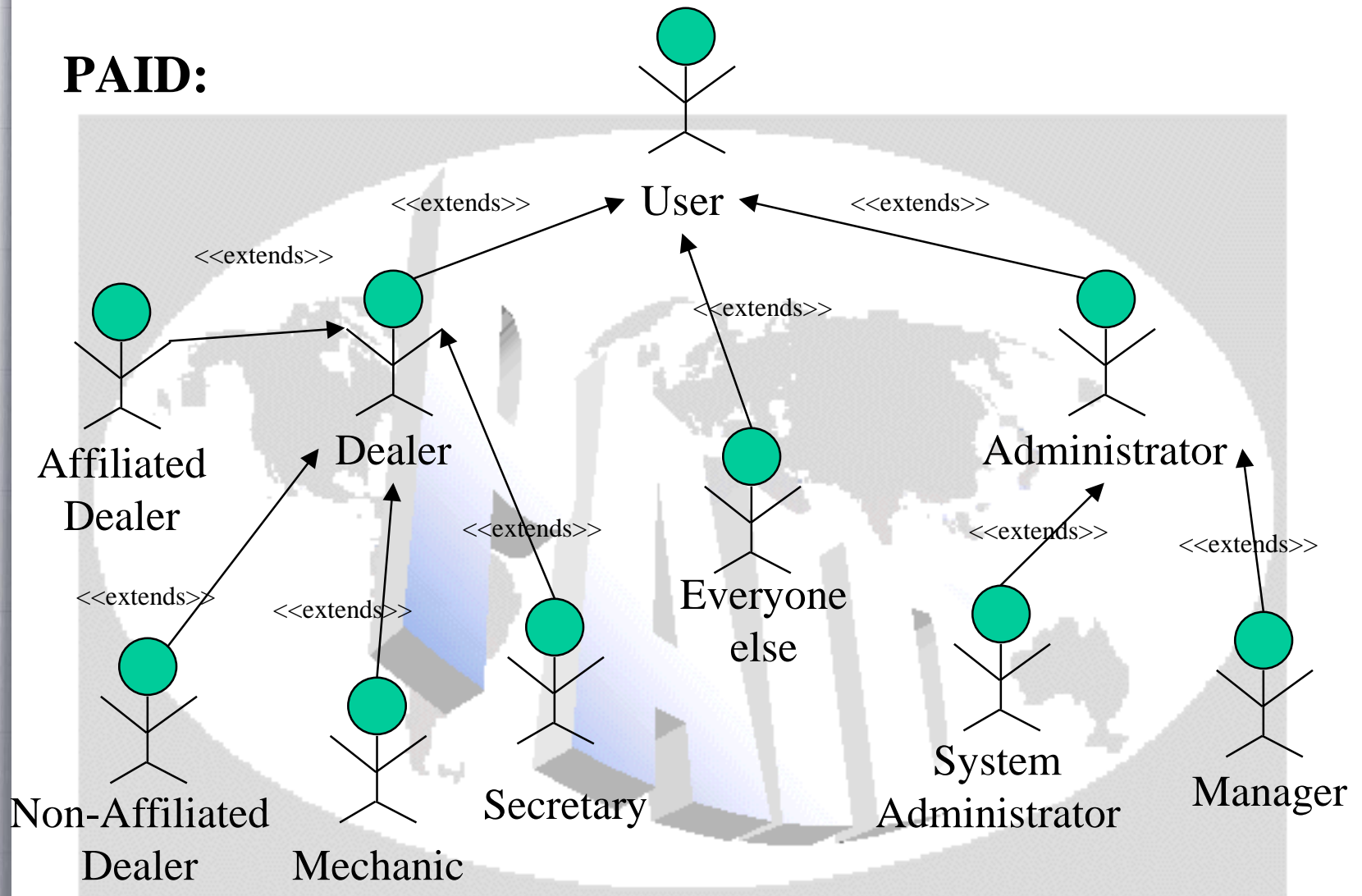
User Model

Authentication:



User Model

PAID:



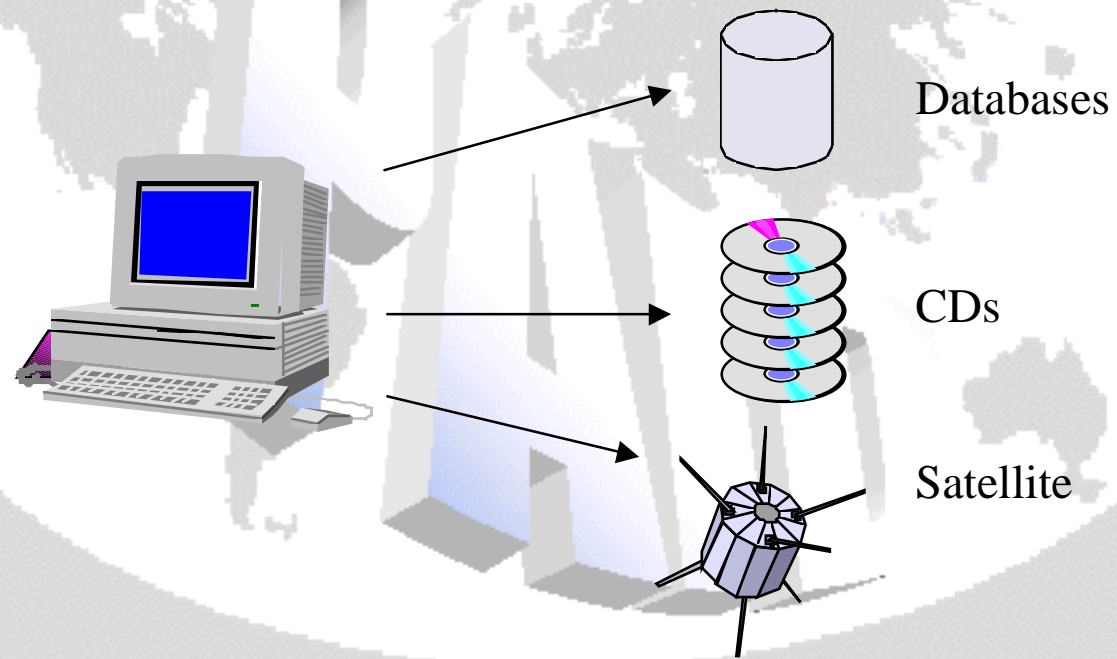


The Smart Card

- The card will be used as a very long password.
- Provides a stronger level of security than regular ID/Password schemes.
- Every authorized user assigned a unique card from Daimler-Benz.
- Required to use the PAID system.
- Each client computer must have a card reader attached.

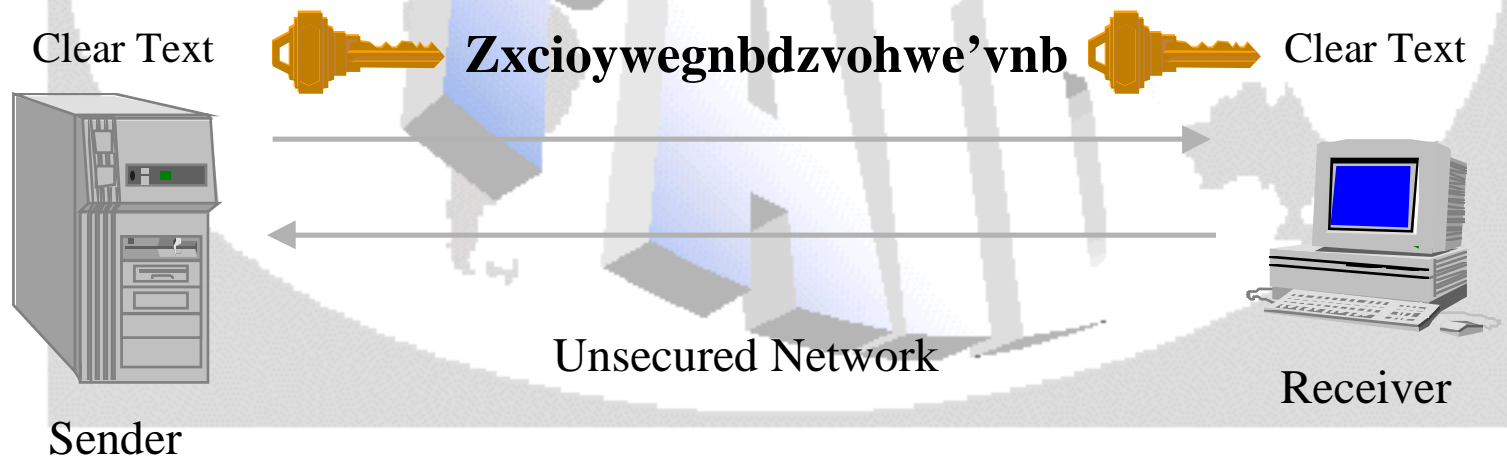
Data Sources

- Authentication will function on all data sources.



Network Communication

- Kerberos, a trusted form of authentication, will be implemented.
- All communication between server and client will be encrypted.



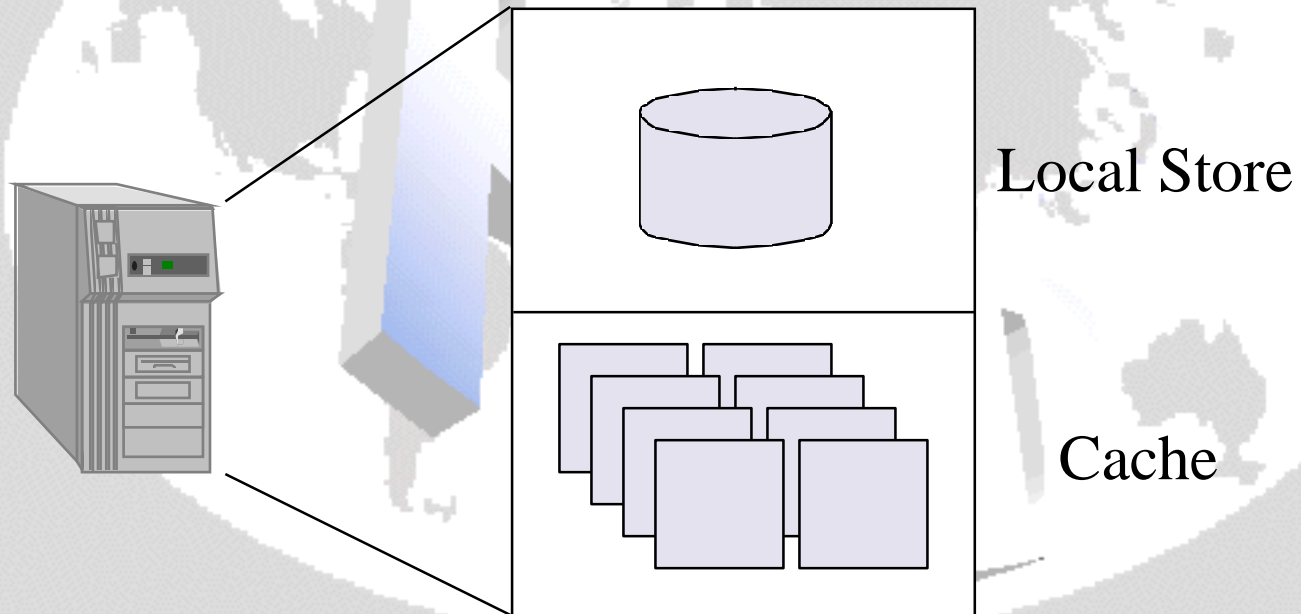
Database - Authentication Interaction

- Authentication will augment the built-in security capabilities of the database.
 - Provides stronger access control.



The Dealer's Server

- Data stored in a local store and a cache.
 - The cache will be stored to make it difficult to read.





Getting Started

- Administrators add a new user to the system.
 - A unique ID and password are created
 - Both are stored on the user's Smart Card
- When the user receives their card, they must activate it.
 - Card activated by either calling headquarters or through the network.
- Users will need to have their smart card inserted into the reader while using the system.



Summary

- A Smart Card will be used for authenticating users to the PAID system.
- The security implementation will be media independent.
- Kerberos will be used to protect transmissions over networks.
- The database system will work with authentication to provide secure access to data.
- Locally, data will be stored in both a local store and a cache.



Section 7

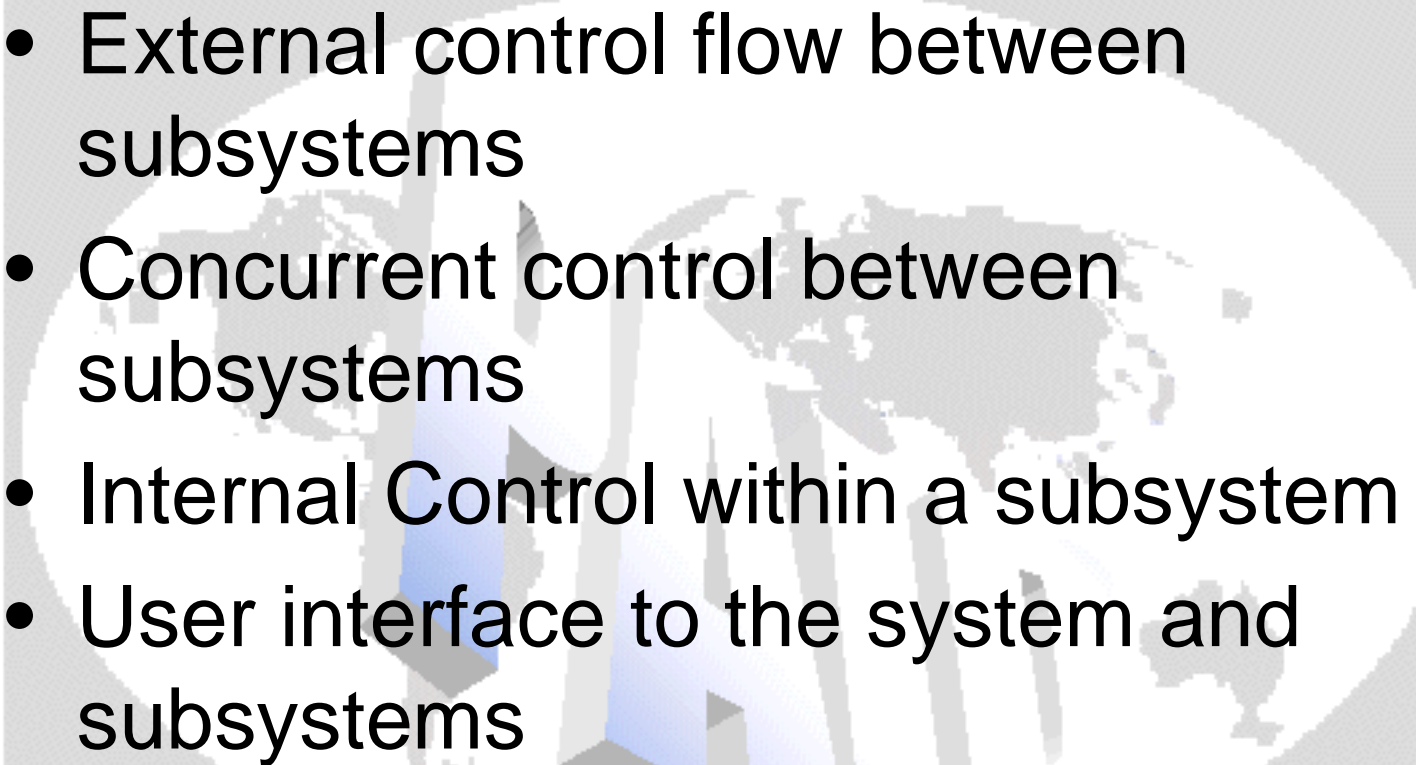
System Control



Presenter: Yun-Ching Lee
Learning Team:
Jonathan Hsieh
James Lampe
Wing Ling Leung
Rudy Setiawan
Jonathan Wildstrom
Andrew Zimdars



System Control - Overview

- 
- External control flow between subsystems
 - Concurrent control between subsystems
 - Internal Control within a subsystem
 - User interface to the system and subsystems

External Control Flow

- **Authentication**

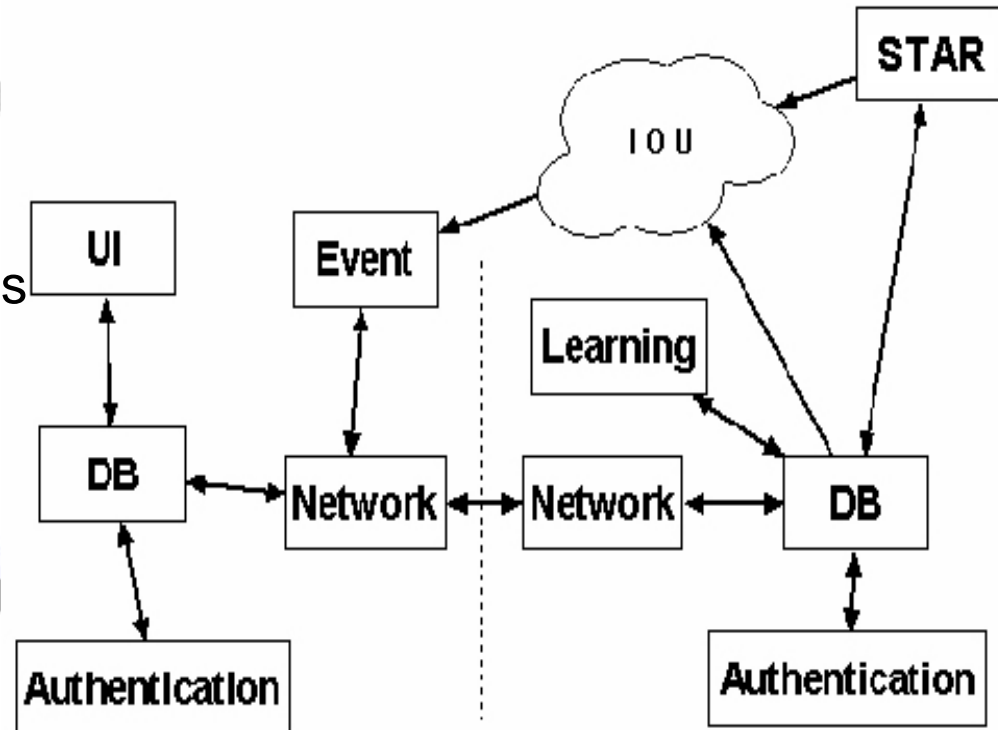
Responds to authentication requests

- **Database**

Responds to queries
May initiate network transfer

- **Network**

Accepts network transfer requests from other subsystems



External Control Flow

- **User Interface**

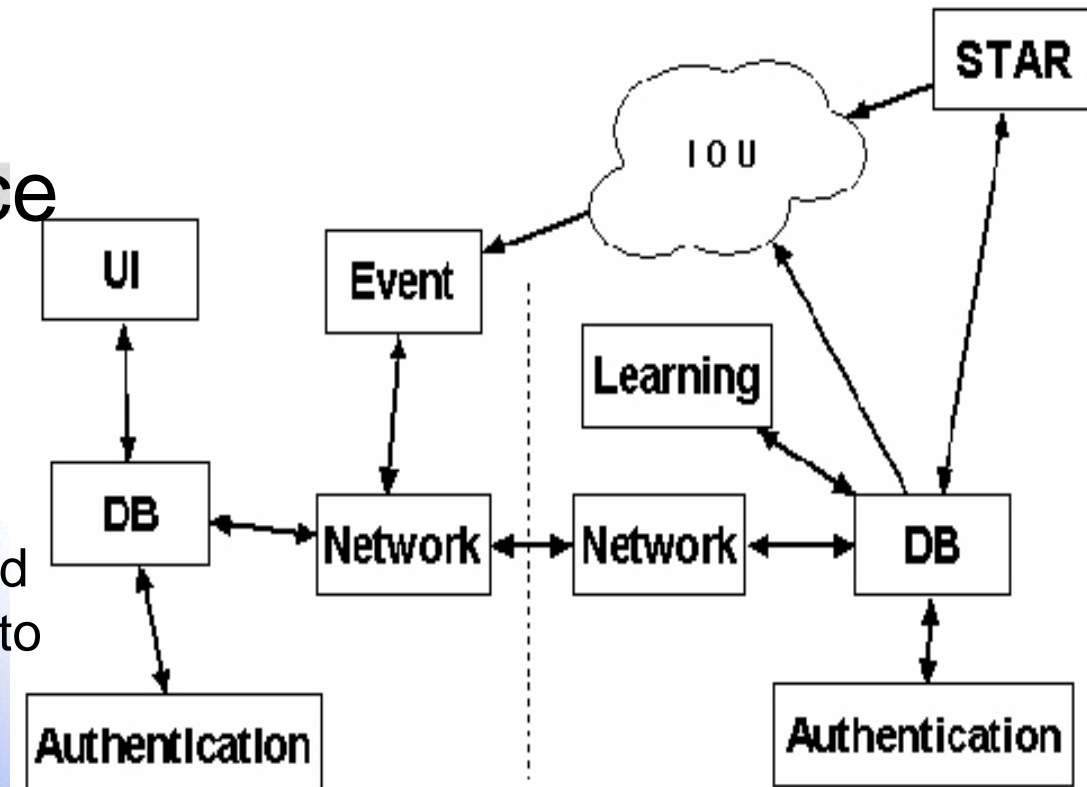
Responds to user inputs
Queries Database

- **Events**

Receives published events and relays to subscribers

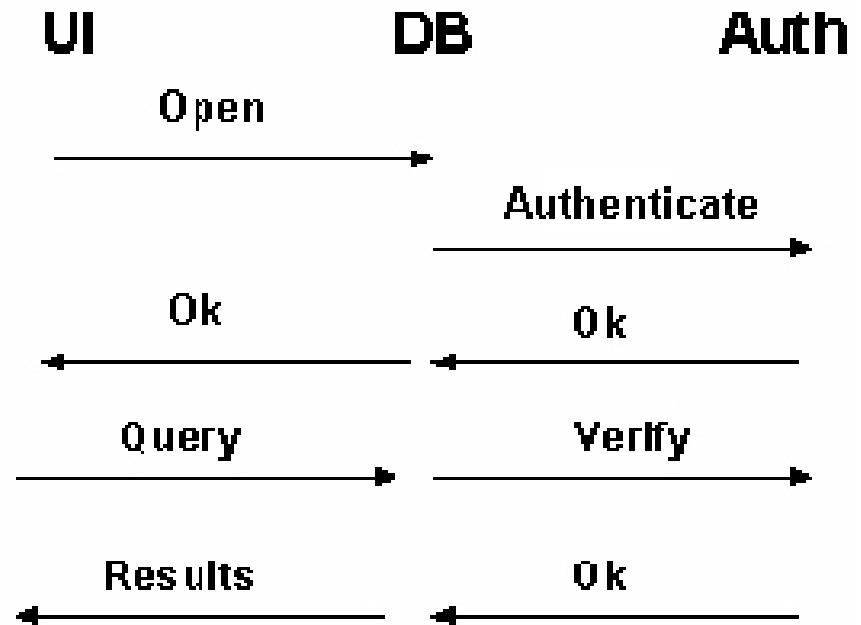
- **Learning**

Responds to Database queries and suggest best action



Concurrent Control

- Subsystem independence allows subsystems to run concurrently
- Ordering of activities prevents normal problems associated with concurrency





Internal Control

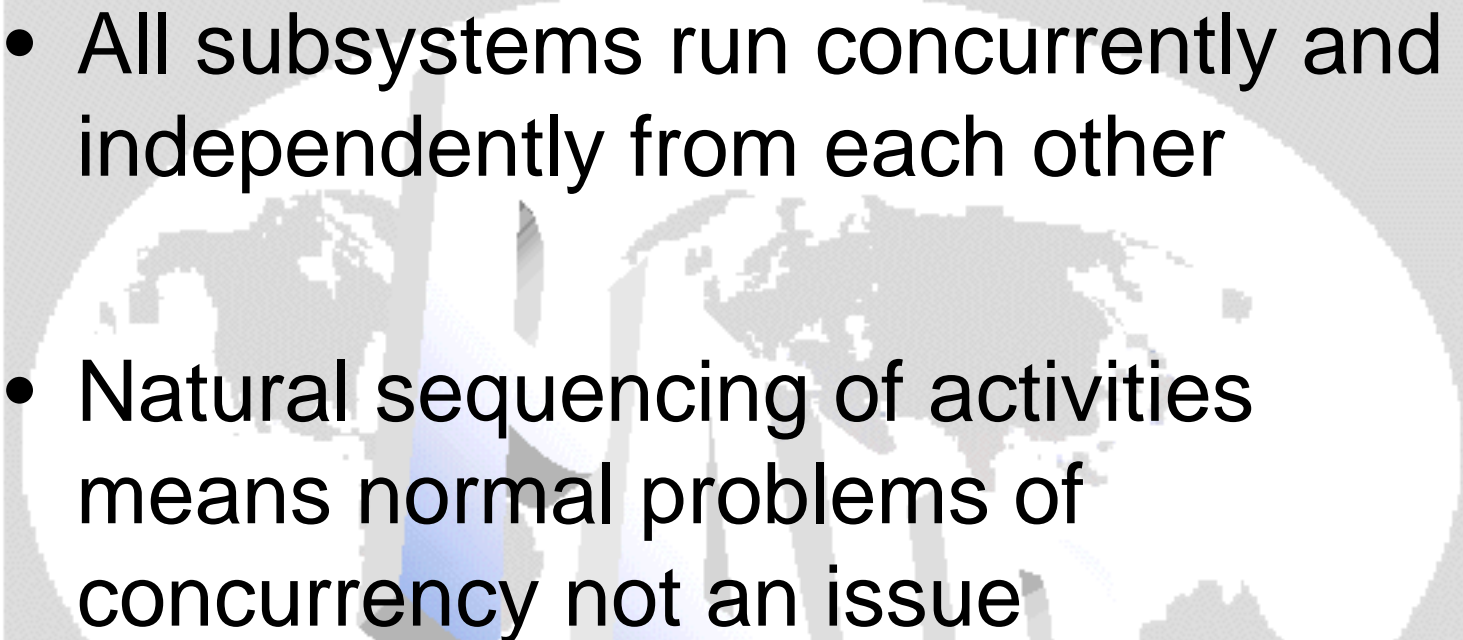
- Internal control flow accomplished by procedural calls
- Event loops for all subsystems
- Learning subsystem scheduler spawns data miner periodically
- User Interface subsystem may spawn input handler threads as needed
- Database locking solves mutual exclusion problem during database updates

User Interface

- User interface for normal user-system interaction is graphical
- Subsystem interfaces for subsystem administration may be graphical or text-based



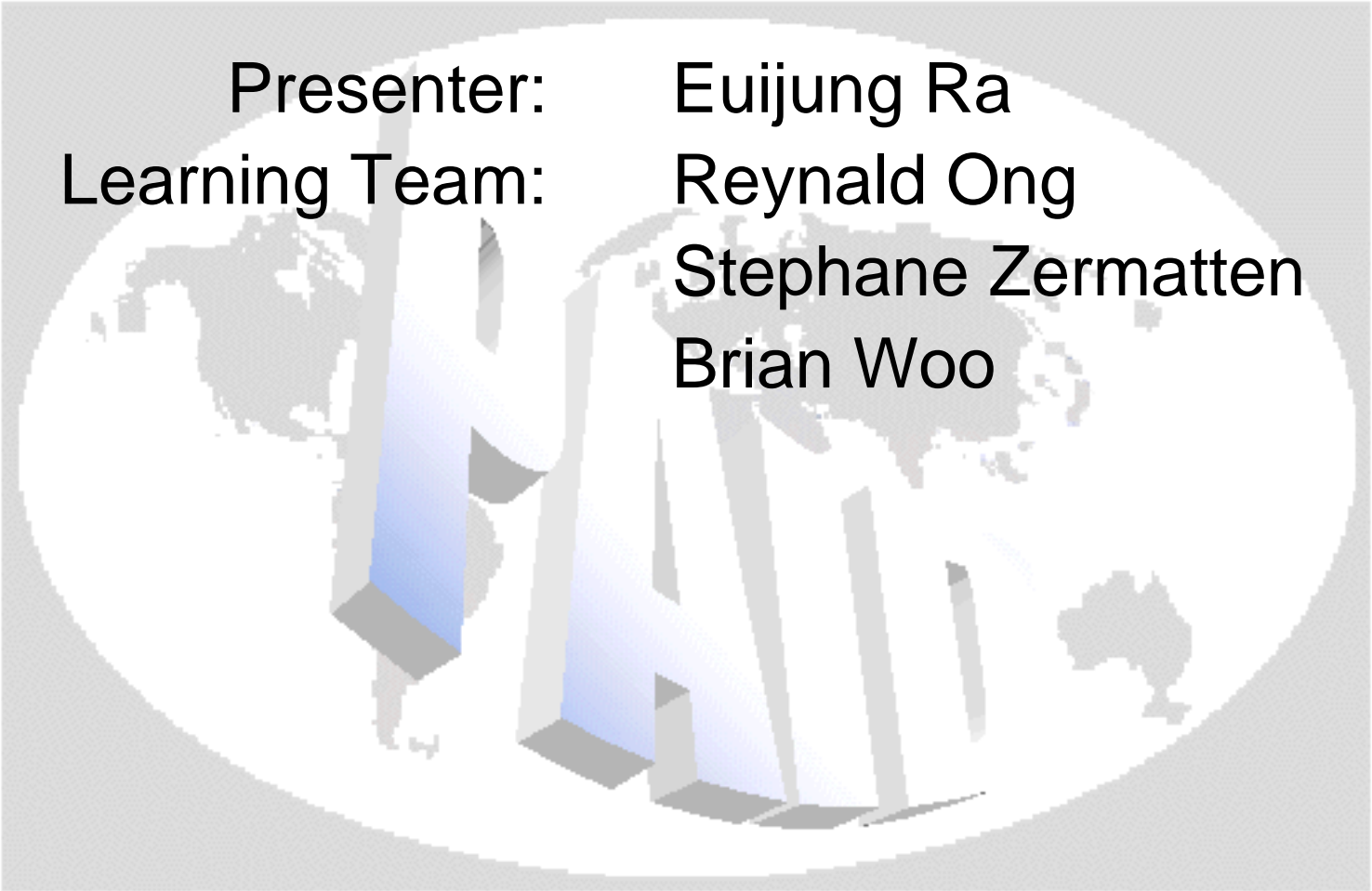
System Control - Summary

- 
- All subsystems run concurrently and independently from each other
 - Natural sequencing of activities means normal problems of concurrency not an issue



Section 8


Boundary Conditions



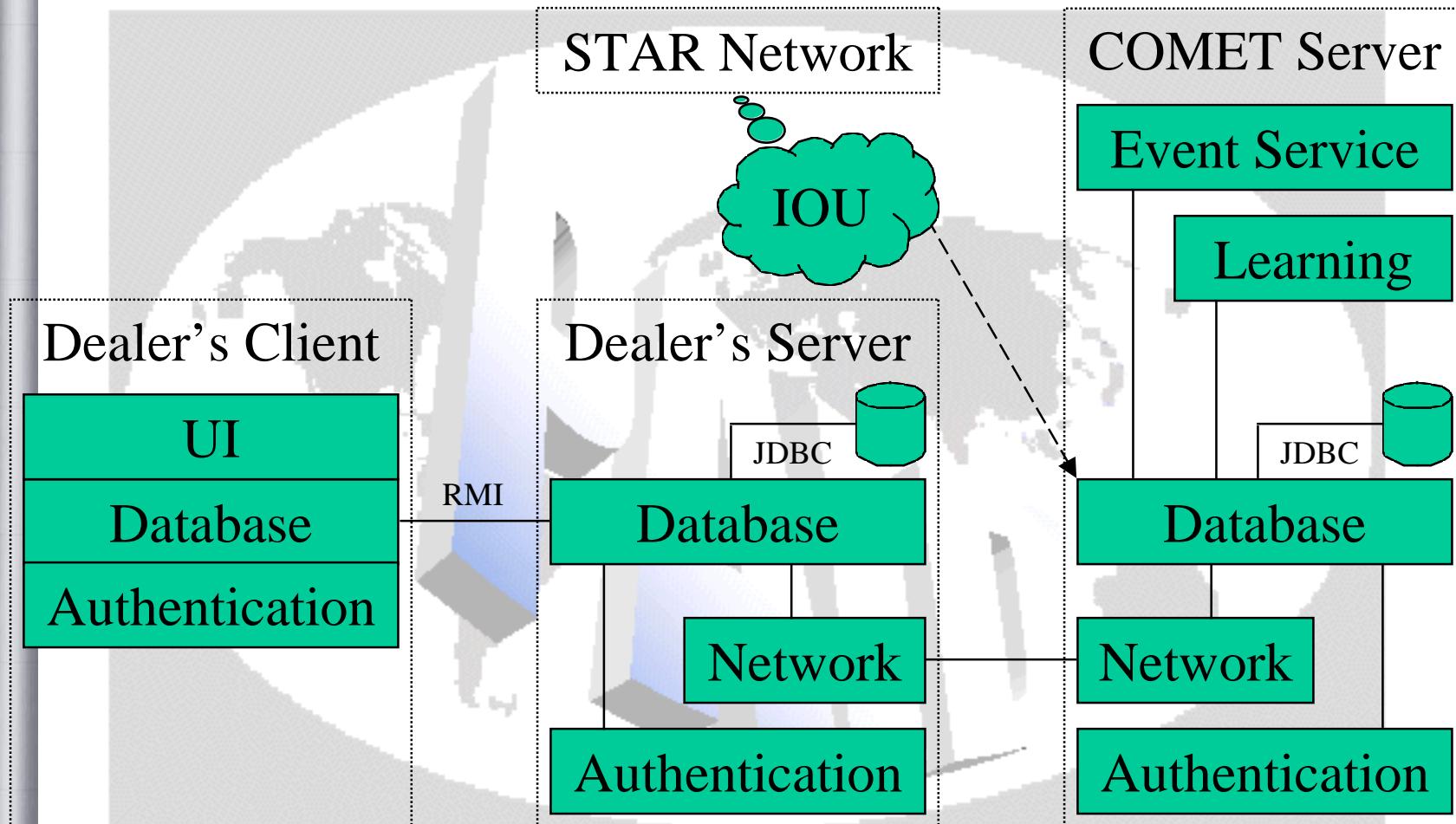
Presenter: Euijung Ra
Learning Team: Reynald Ong
Stephane Zermatten
Brian Woo



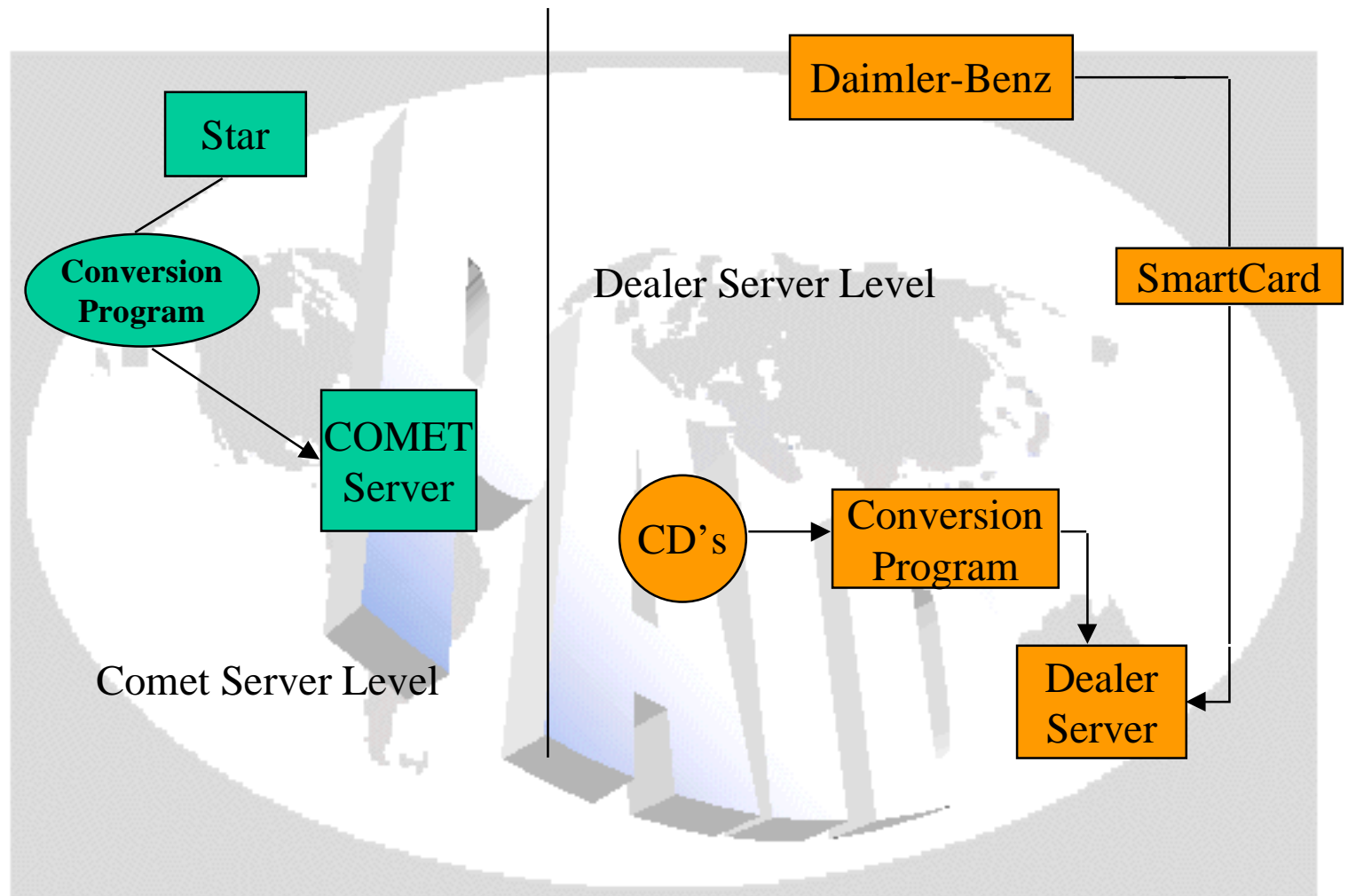
Outline

- 
- Initialization
 - Termination
 - Failure
 - Screenshots

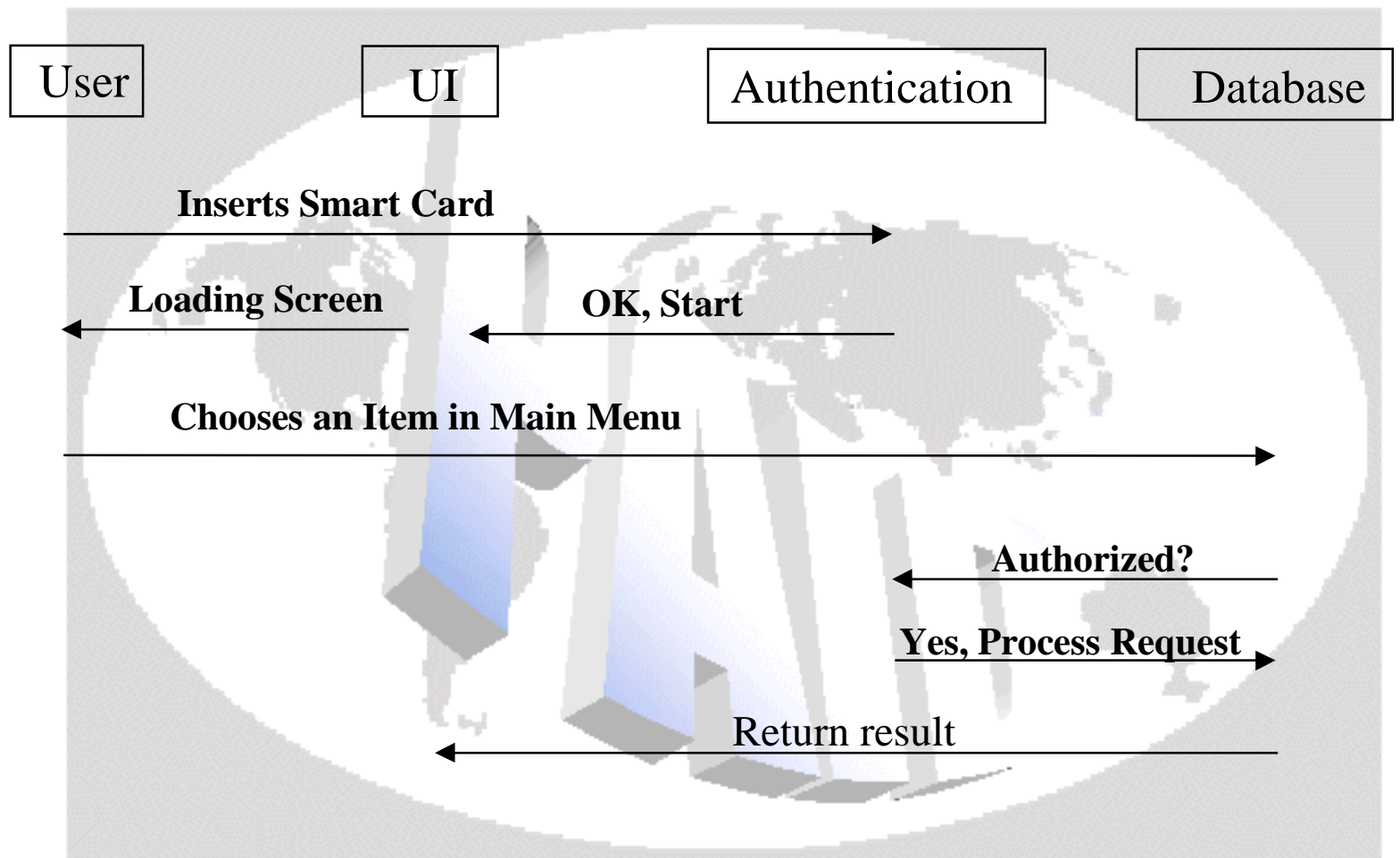
Deployment



Initialization of the Whole System

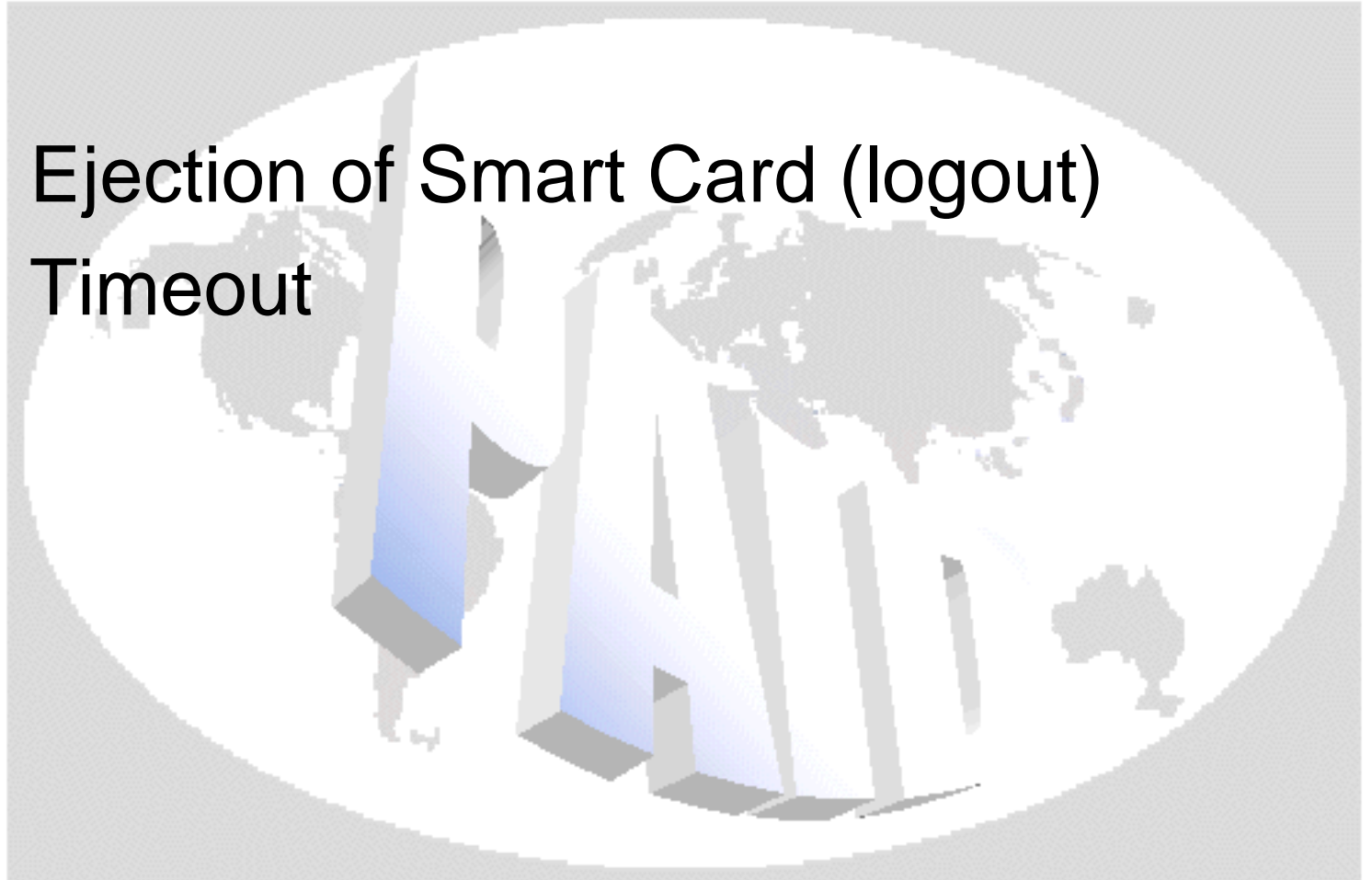


Start-up of a User Session



Termination at User Level

- Ejection of Smart Card (logout)
- Timeout

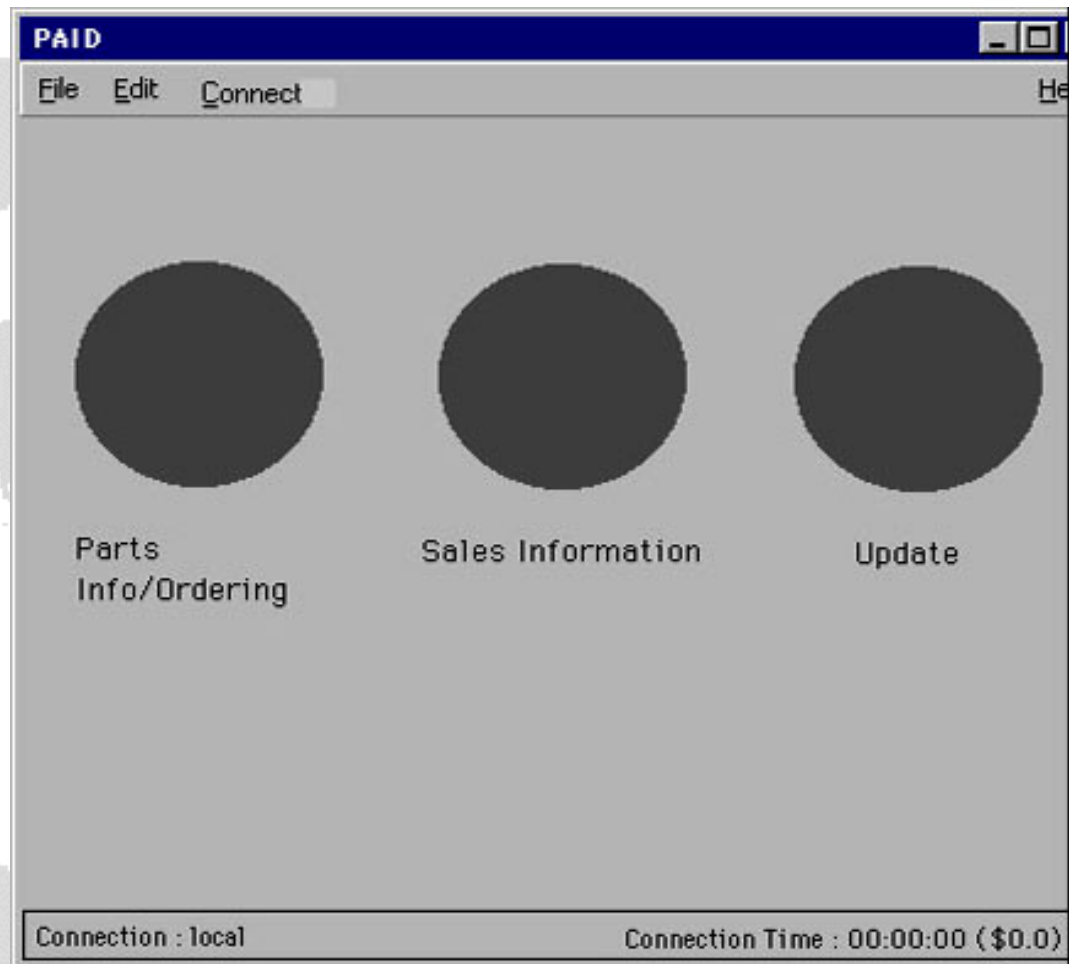




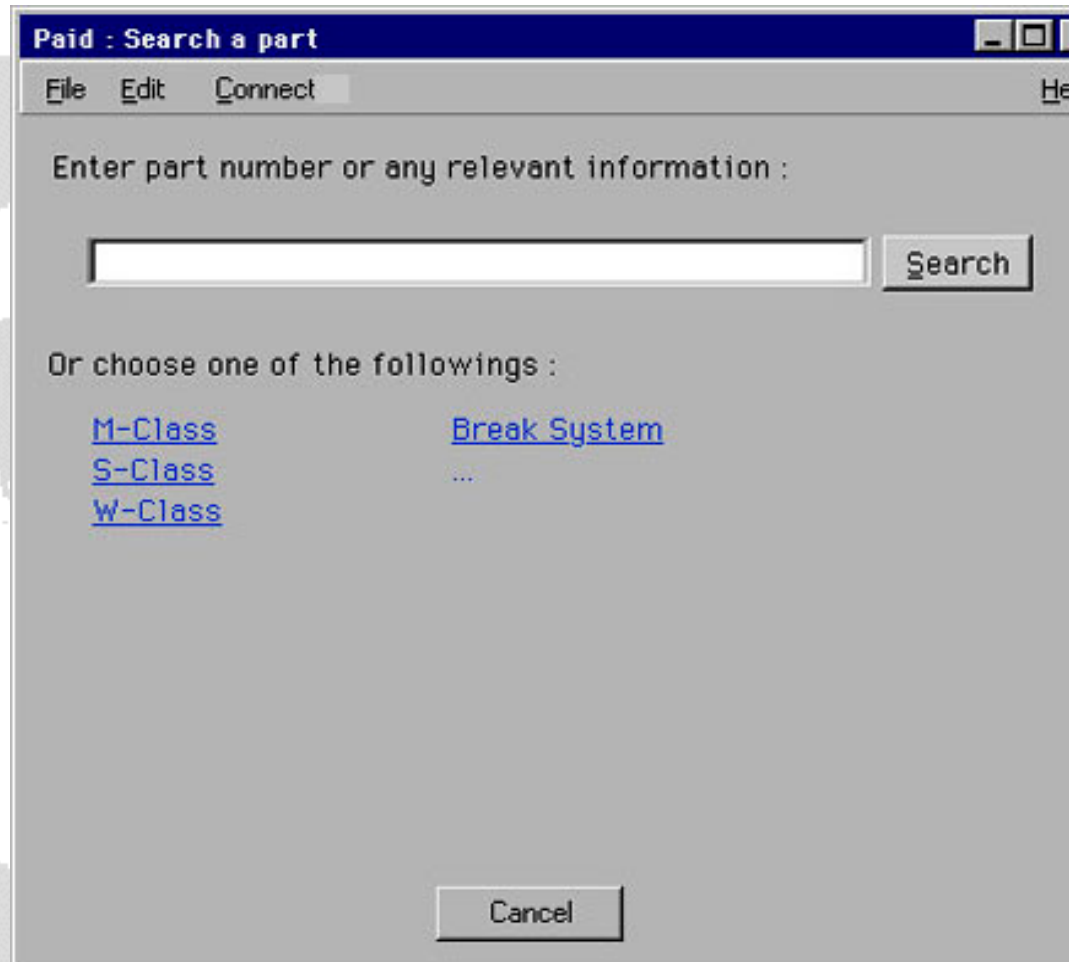
Failure

- Network Server Link - reroute (alternate fall back link)
- Dealer LAN - local autonomy
- Local Database - possible recovery facilities provided via saved updates (special server)
- Server Database
 - Mirroring and replicating combined with hot incremental backups.
 - Should permit continuous availability.

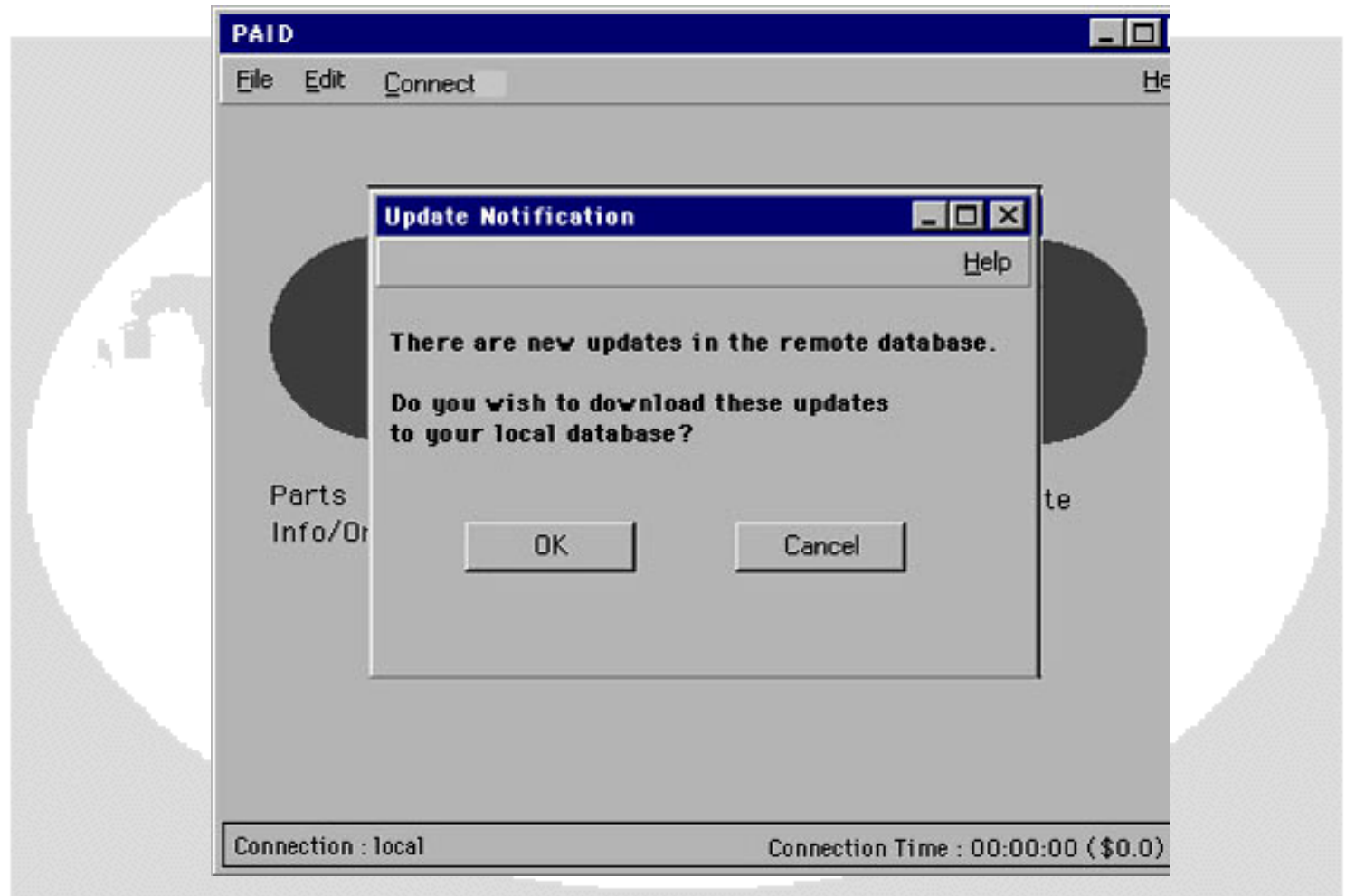
Screenshots



Screenshots



Screenshots




Screenshots






Section 9

Design Rationale



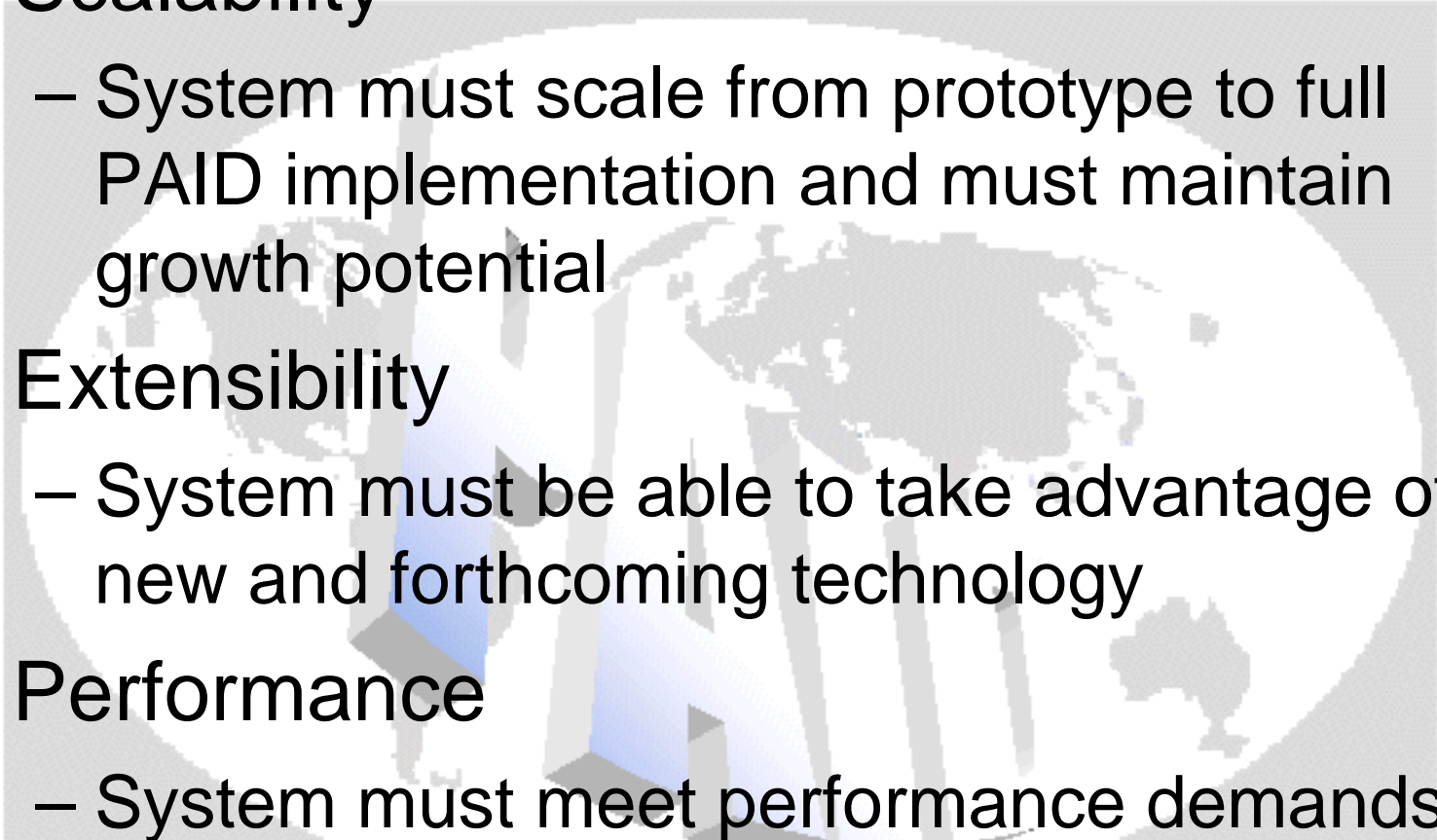
Presenter: Ivan Tumanov
Learning Team: Georgios Markakis
Richard Markwart
Timothy Shirley

Outline

- Goals
 - Issues
 - Alternatives, Criteria and Rationale
 - Summary
- 

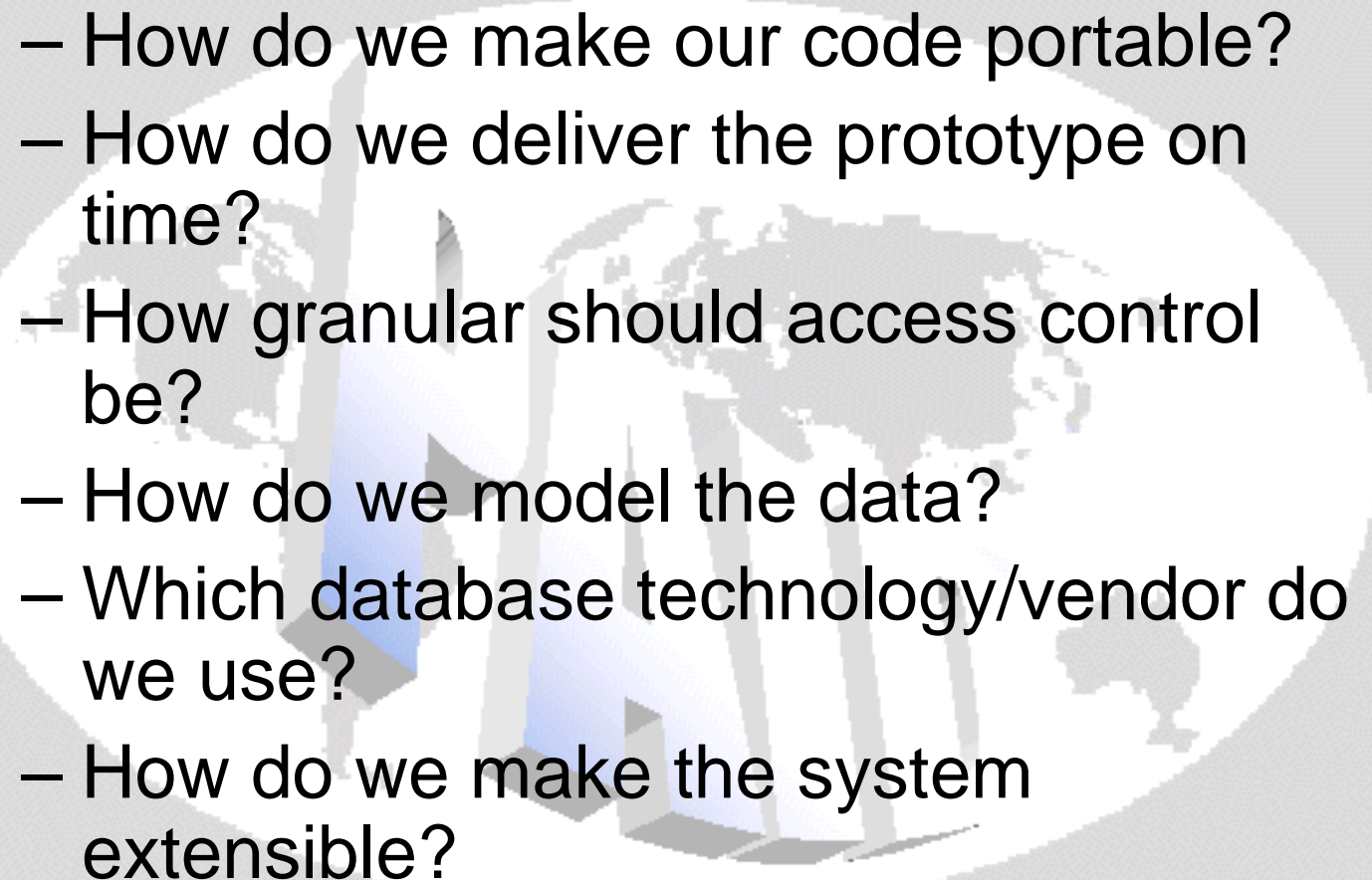


Goals

- Scalability
 - System must scale from prototype to full PAID implementation and must maintain growth potential
 - Extensibility
 - System must be able to take advantage of new and forthcoming technology
 - Performance
 - System must meet performance demands
- 

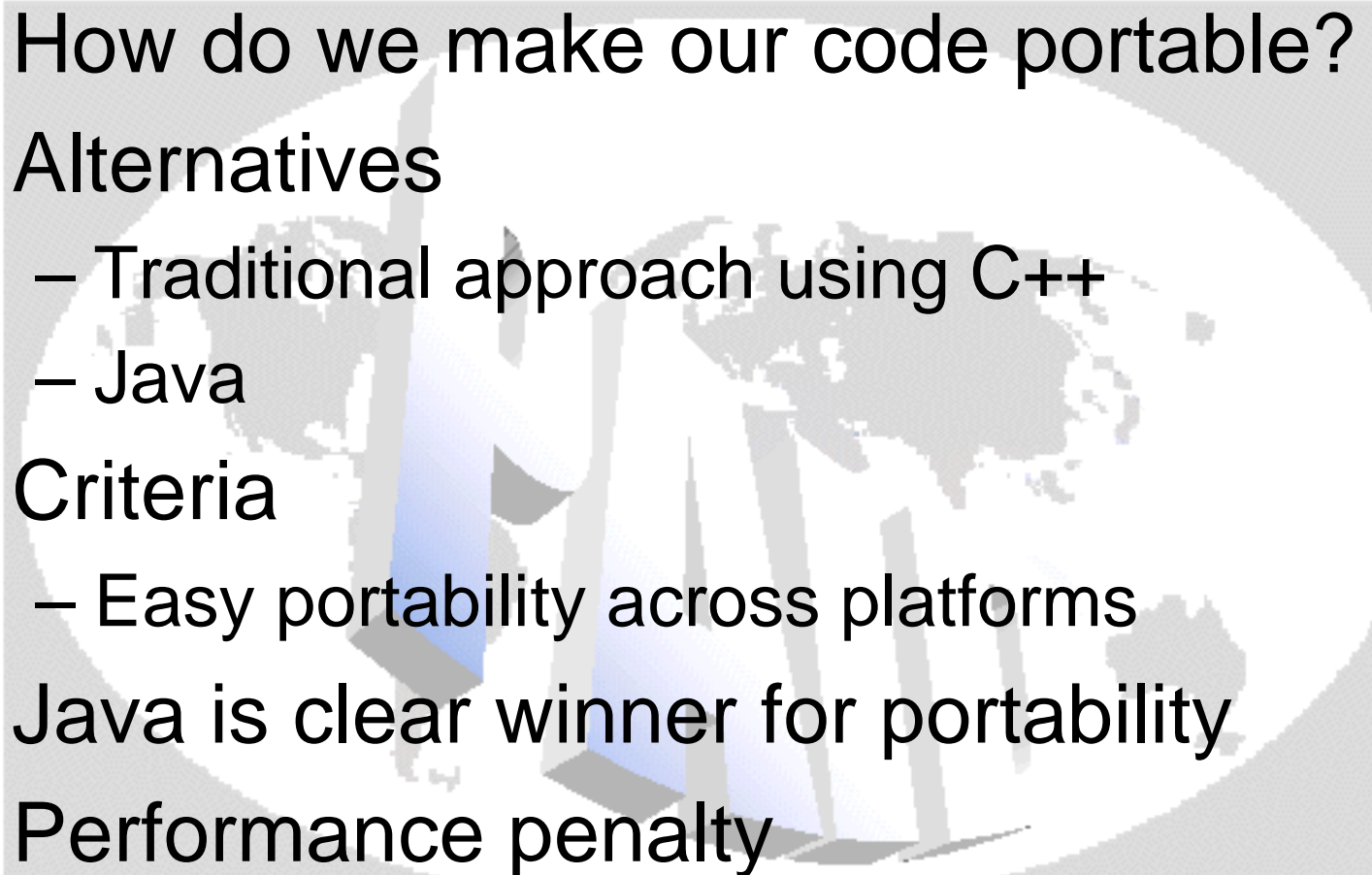


Issues

- 
- How do we make our code portable?
 - How do we deliver the prototype on time?
 - How granular should access control be?
 - How do we model the data?
 - Which database technology/vendor do we use?
 - How do we make the system extensible?



Platform Portable Code

- How do we make our code portable?
 - Alternatives
 - Traditional approach using C++
 - Java
 - Criteria
 - Easy portability across platforms
 - Java is clear winner for portability
 - Performance penalty
- 

Time to Delivery

- How do we deliver the prototype on time?
- Alternatives
 - Incomplete prototype
 - Limit scope of prototype
- Criteria
 - Hard deadline of semester length
- Concentrate on key points of functionality
- No large-scale network activity (6000+ machines), no billing, no migration from machine to machine



Access Granularity

- How granular should access be?
- Alternatives
 - User-based security only
 - Record-based security
 - Data subset/table based security
- Criteria
 - Efficient, Secure access
- Data subset/table based security

Simple Data Model

- How do we model the data?
- Alternatives
 - Analysis and redesign of full data model
 - Current data model for STARNetwork
 - Simple Data Model
- Criteria
 - Expediency, efficient prototype
- Simple Data Model to demonstrate IS requirements, more thorough treatment later

Database Vendor Independence

- Which database technology/vendor?
- Alternatives
 - Specific vendor technology (Oracle, Sybase)
 - Middleware technology (ODBC, JDBC)
 - Relational vs. Object Oriented vs. flat file
- Criteria
 - Should not be tied to a particular vendor
- JDBC and a data abstraction layer
- Performance Hit

Extensibility

- How do we make the system be extensible?
- Alternatives
 - Use proven, fully developed technologies
 - Use new technology, anticipating changes
- Criteria
 - Don't limit design to lowest common denominator of what's available today
- Design system to take advantage of current and future technology

Summary

- Design Time Window considered
- We are on the threshold of
 - Easier, cheaper machines
 - Chip-based Java
- Commonly available PDAs
 - Proprietary technology powerful enough to run PAID exists today
 - High performance PDAs are becoming commonly available
- Design Rationale to support
 - Scalability, Extensibility, Performance
 - Implemented using good Software Engineering Principles

PAID System Design

