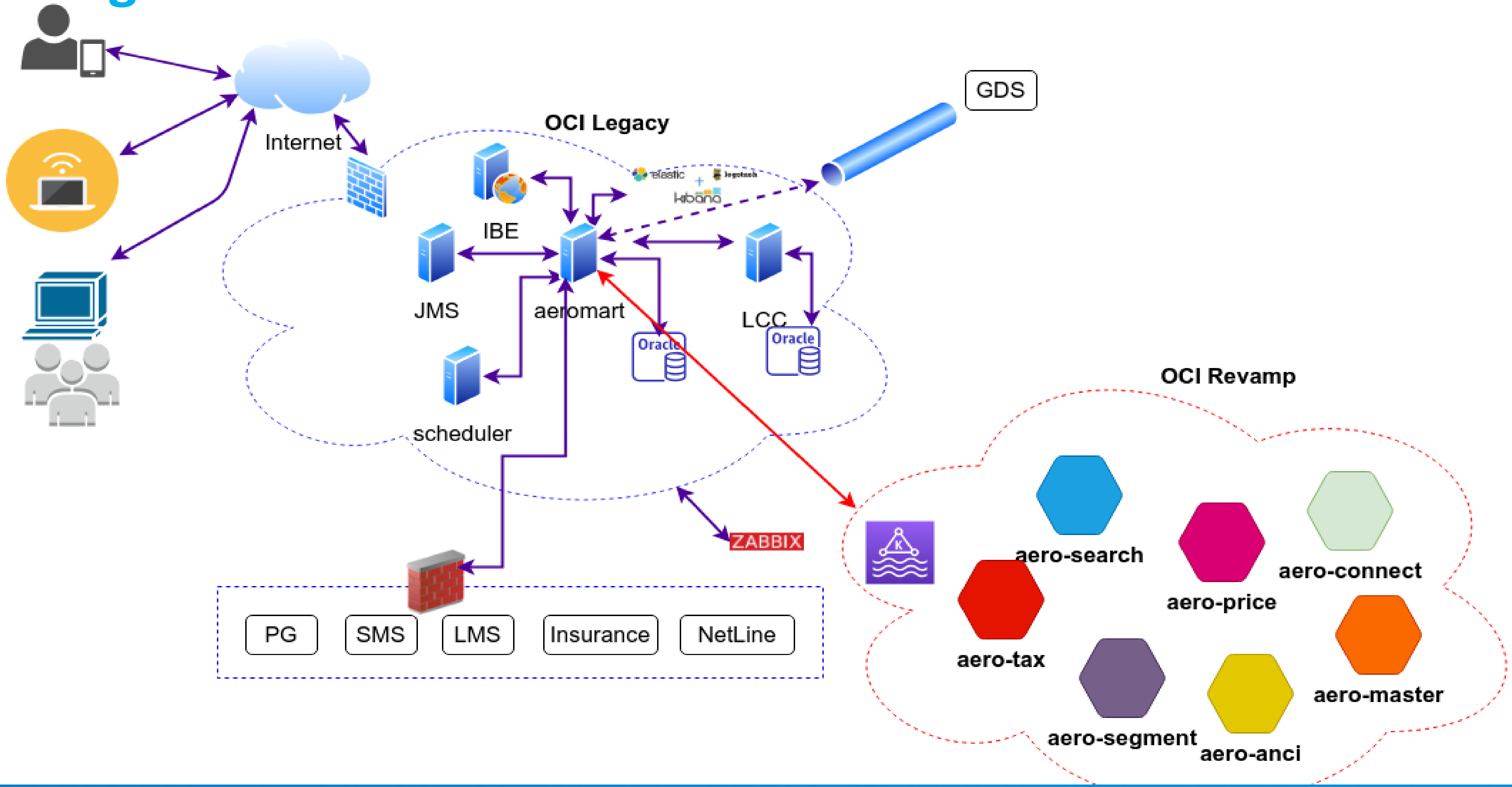
AeroMART Technical KT

High Level Architecture



Technology Stack 1

- **Java 8**
- **Oracle 12c/11g**
- **Struts2/ SpringMVC** - For implementing MVC in presentation tier
- **Hibernate (3)** - For implementing ORM
- **Spring JdbcTemplate** – for Complex Queries
- **Spring (4.25)** - To manage modules and modules' configurations through IoC, Uses the AOP engine for implementing cross-cutting concerns
- **EJB3** – Statless Session Beans/ MDB

Technology Stack 2

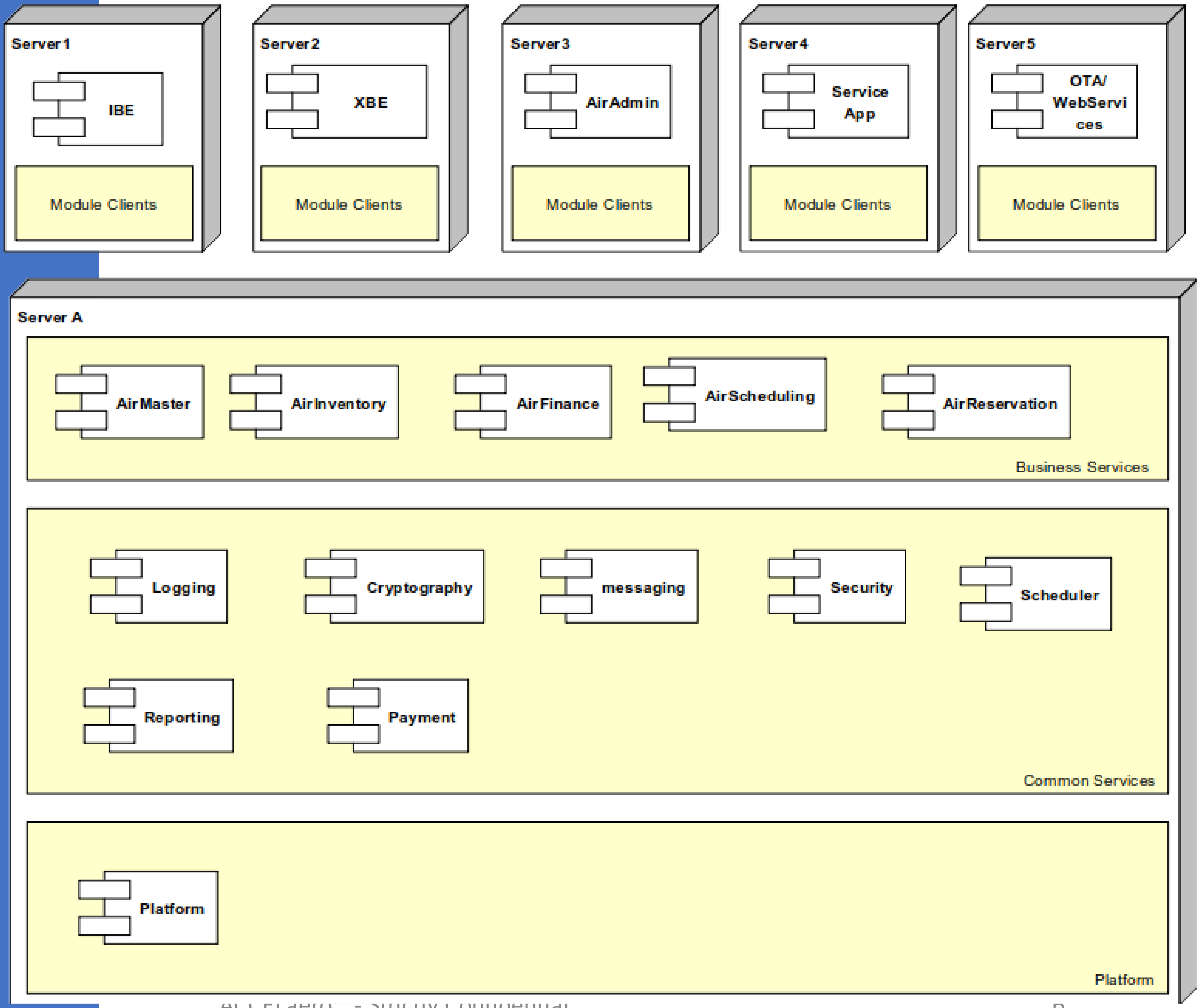
- **Quartz (2.2.1)** - Scheduling engine
- **Jasper Report (4.7.0)**– For reporting template managements
- **JUnit** - For unit testing functionalities
- **Velocity** - For templated email/sms message generation
- **xDoclet** - For generating EJB interfaces and hibernate mappings
- **JavaCC** - grammar specification and converts it to a Java program that

can recognize matches to the grammar. (ETL, PFS, PNL, ADL, SSM, ASM, PRL, AVS)

Technology Stack 3

- **Redis** – cache frequently accessed data (WebServices/OTA, Bundle)
- **Aerospike** –
- **Ant** - Build automation and unit test automation
- **Jboss** (4.2.3.GA)
- **Docker** – Containerization
- **Frontend – AngularJs (1.*)/Jquery/JSON/Ajax/Javascript/Html/css**

Moduler Architecture



Moduler Architecture

Frontend Services	aaservices/ibe/service-app/webservices/xbe
Webplatform	webplatform
Business Services	airreservation/airprice/airschedule/travelagent/airinventory
Common Services	invoicing/messaging/reporting/scheduler/paymentbroker/ login/messagepasser/airsecurity
Platform	platform

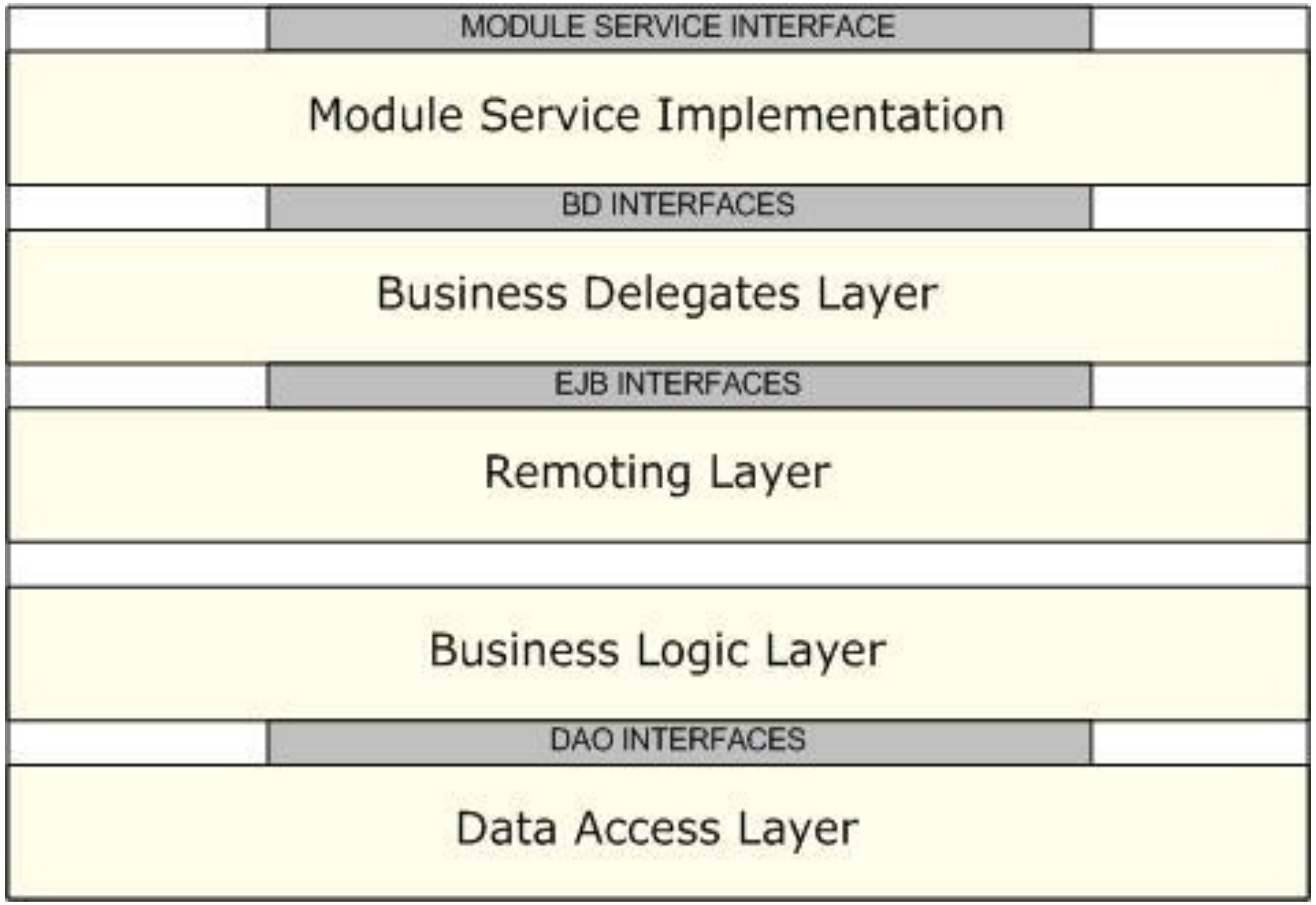
Key Strategies

- A key concept in developing flexible and extensible architecture is strong encapsulation of coherently related functionalities into logically separable units – modules
- ISA platform provides services to build and manage modules
- **Layering & Controlled specialization**
- The elements with constrained dependencies are grouped into layers. The upper-level layers depend only on the lower-level layers. The controlled specialization enforces the concept of defining common services at lower-level layers while specialized services per vertical/deployment are defined at the higher-level layers.
- **Modularization**
- Key to scalable and usable architecture. A module provides a well managed and well defined set of service interfaces and contracts to the application.

What is a module?

- A large grained unit of software; Encapsulates coherently related functionalities into logically and physically separable unit
- System architecture is described in terms of modules and their interactions
- Each module has its own configurations, source and build
- Client/Carrier wise module configurations
- A module may share globally defined configurations as well
- Module has a well defined service interface through which it exposes services to clients;

Layering with in Module



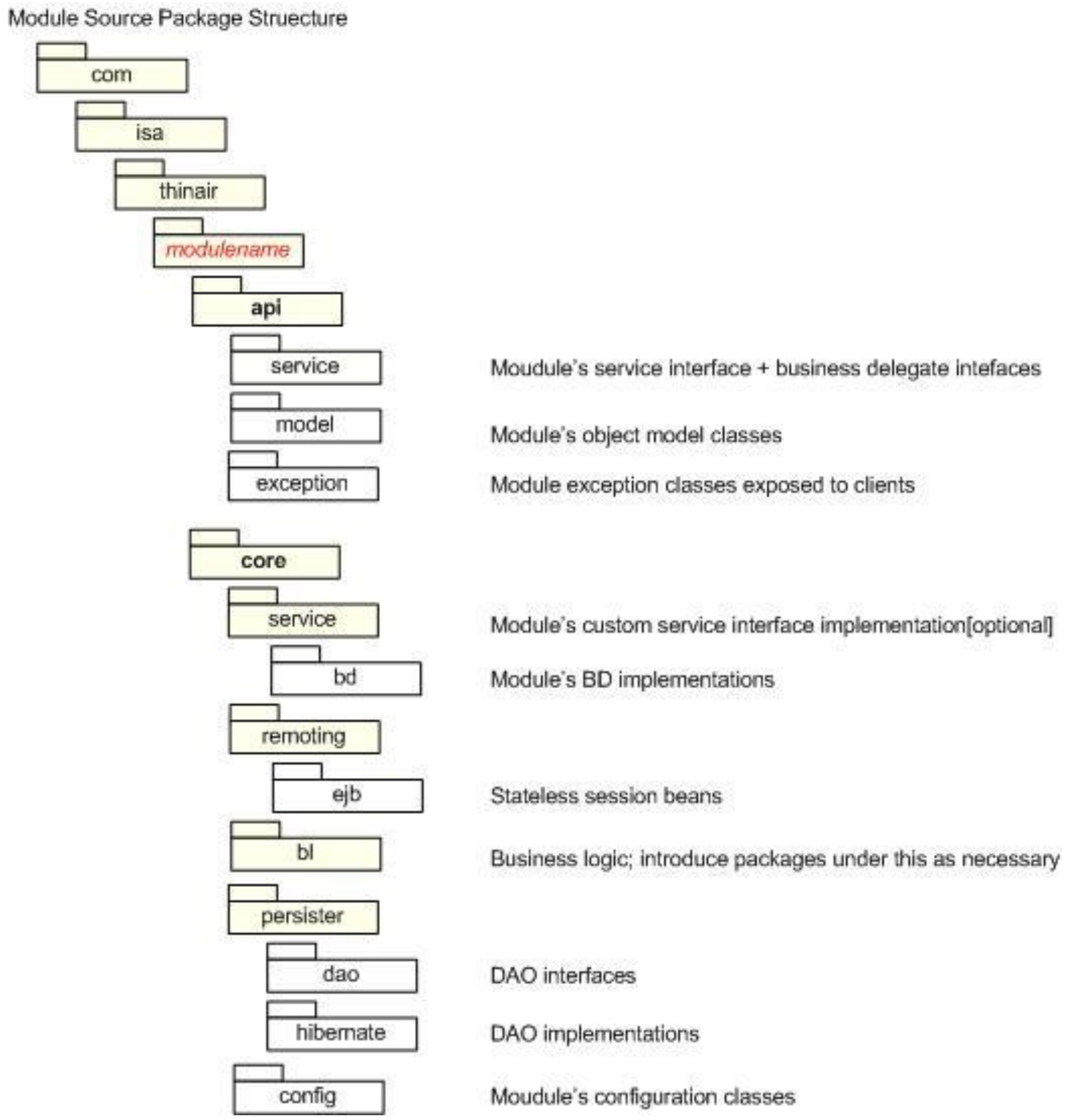
Layering with in Module

- Data Access layer takes care of data persistence
- Key business functionalities are implemented in business logic layer
- Remoting layer wraps business logic in stateless session beans; both locally and remotely accessible
- Business delegates layer provides location transparency
- Module’s service implementation exposes the business delegates

Module's Build

- Apache Ant is used for build automation and unit test automation
- Module build is defined reusing the generic targets defined in the global build file – modulebuildutils.xml; Module’s build may override globally defined build properties to customize the its build
- Third party libraries needed for the module build are referred from <project-root>/repository/lib
- Module distributions are copied to <project-root>/repository/modules so that other modules can share
- Refer comments in modulebuilddepends.xml and modulebuildutils.xml for further information

Module's Java Source Packaging


Module Source Package Structure
com
isa
thinair
modulename
api
service
Moudule's service interface + business delegate intefaces
model
Module's object model classes
exception
Module exception classes exposed to clients
core
service
Module's custom service interface implementation[optional]
bd
Module's BD implementations
remoting
ejb
Stateless session beans
bl
Business logic; introduce packages under this as necessary
persister
dao
DAO interfaces
hibernate
DAO implementations
config
Moudule's configuration classes

EJB3 & JMS

- Mainly Spring IOC features are used for injecting dependencies and configurations through spring bean factory configurations

- Stateless Session Beans are used for transaction injection & transaction demarcation and for remoting; Failover and load balancing features are also used in clustered environment

- Message Driven Beans along with JMS queue provider comes with JBoss are used for asynchronous processing

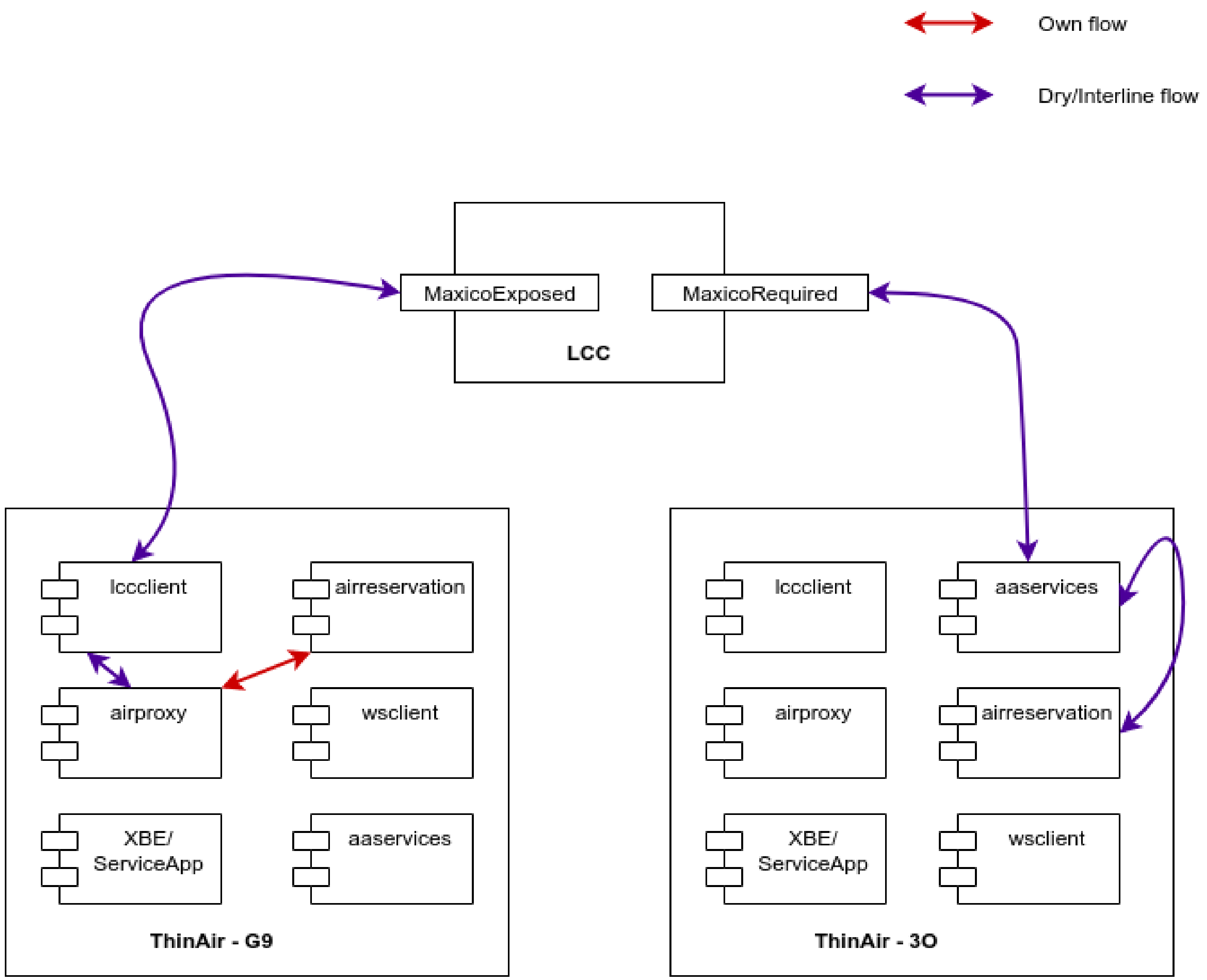
Webservices

- AccelAero has two WS implementations – XFire & JAX-WS
- Services exposed for OTAs are implemented using XFire whereas Services for LCCONNECT interaction are implemented using JAX-WS
- Both uses schema first approach and JAXB2 does the schema to Java classes generation and serialization & de-serialization of the messages between XML and JAVA objects

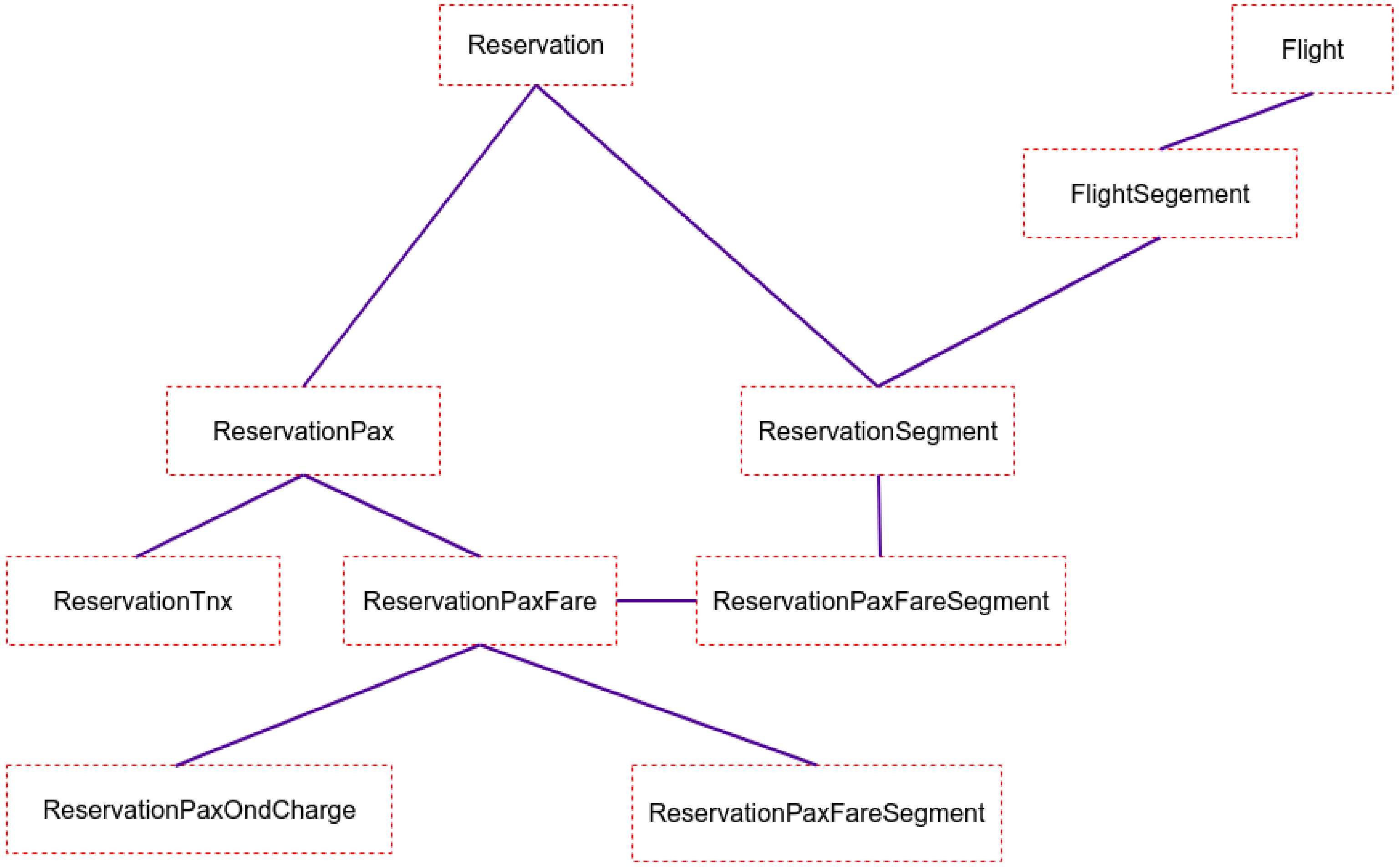
Why LCC ?

Introduction to LCC

- Agreements
- Routes
- Exchange Rates
- Interline Fares



Reservation main entities



Enhancements/ Milestones 1.

- Version – 1 vs 2
- Library upgrades - (Java, Spring, Hibernate, Jasper, oracle driver)
- LCC - Dry/Interline=> 2009-2011
- Repository => CVS -> Git
- Requote - 2013 - 2014
- Mahan CR – 250/300
- Redix Cache => WebService
- Single Code base – 007.0

Enhancements/ Milestones 2.

- DCS Connectivity
- GDS/Codeshare support
- Service App
- Netline
- Containerized/Docker/Automation => Cluster to Hybrid
- Cloud Migration

Recommendations

- Rebasing the feature branch frequently
- Take update from the remote daily
- Use formatter and configure according to the IDE
- Code reviews are mandatory and if required do peer review
- Check Other areas as well – Reference/JIRA (Old/New)
- Avoid unnecessary formatting or refactoring
  - Functionality may break, Requires further testing.
  - Don’t push local config changes
  - Merging with release branch and rebasing will leads to conflicts.

Revamp Integrations - P0

- P0 Integration - (aero-segment)
  - Bundles
  - Baggage
  - SUR charges

Revamp Integrations - P1A

- P1A Integration -
  - aero-price – OnHold Configurations
  - aero-connect – SSM/ASM
  - anci-promotion - promotions

Revamp Integrations - P1B

- P1B Integration -
  - aero-search
  - aero-price
  - aero-tax
  - aero-surcharges
  - aero-master
    - Sync services –

      Currencies & Exchange Rates
    - OND.JS

Revamp Integrations - P2

- P2 Integration
  - aero-agent
  - IFG
  - aero-suite
  - Sync services – Agents, Users & Invoices

Revamp Integrations - P3

- Aero-Order Integration
  - Parallel Mode
  - Syn mode - TODO
  - Full Cut-over - TODO

- Aero-Anci Integration
  - Bundles
  - Ancillaries

Questions ?

References

Old Confluence
https://confluence.isaaviations.com/

Old Jira -
https://jira.isaaviations.com

https://jiraisa.atlassian.net/wiki/spaces/AI/overview?homepageId=120488072

Thank you