PRODYNA
information architects

09.07.2014

ACRES (AirCraft REservation System) Architectural Overview

Martin Bisanz, PRODYNA AG

© PRODYNA AG 2014

PRODYNA
information architects

- **Overview**
- Demo
- Coding
- Non-functional requirements
- Feedback

© PRODYNA AG 2014

SCM / CI

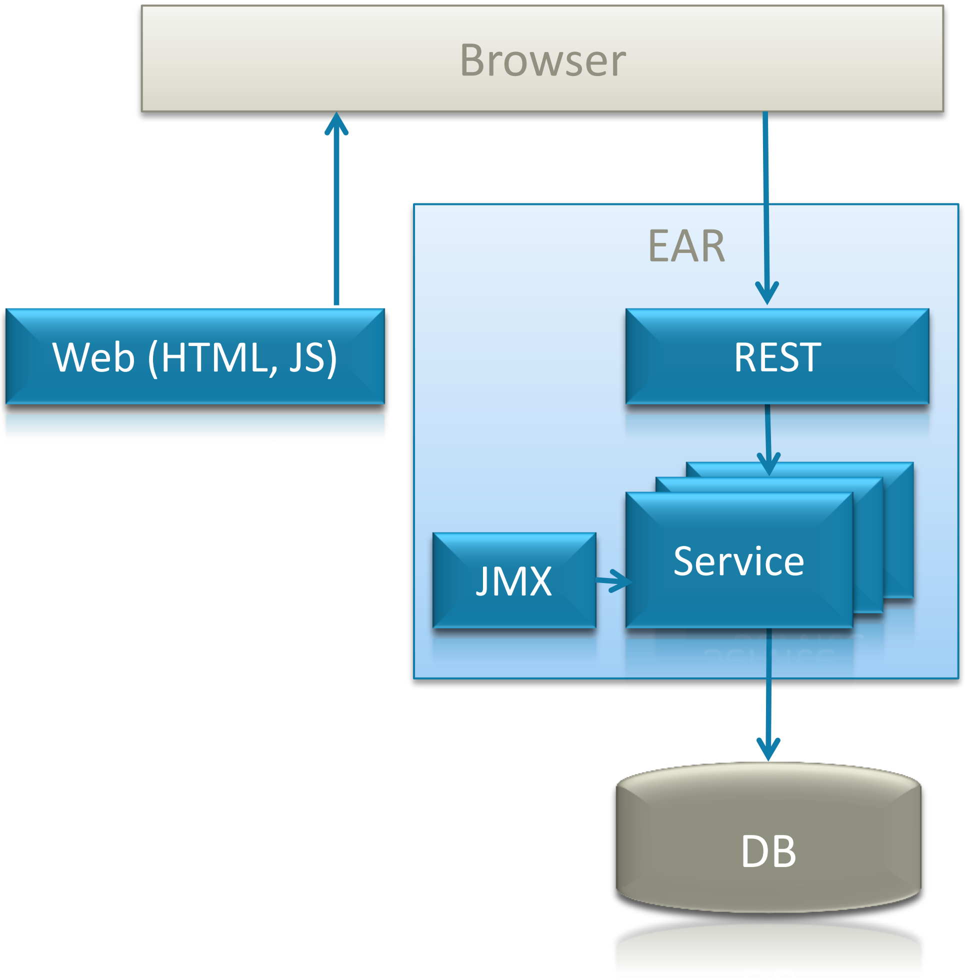
- GitHub: https://github.com/mbisanz/acres.git
- Pushing to upstream only
  - if everything compiles
  - if all tests pass
  - When changing the client (acres-web), re-build before push

- BuildHive: https://buildhive.cloudbees.com/job/mbisanz/job/acres/
- CI Cloud Server (Jenkins)
- Integrated with GitHub

© PRODYNA AG 2014

Application Architecture

- Java EE 7 / JBoss
  - Web Front-End
    - HTML/JS
  - REST Layer
  - Service Back-End
  - JMX Monitoring



Technologies

Back-End

- Java EE 7/JBoss WildFly 8
- EJB, CDI, JAX-RS
- JPA, MySQL
- JMX

Front-End

- HTML
- JavaScript / AngularJS

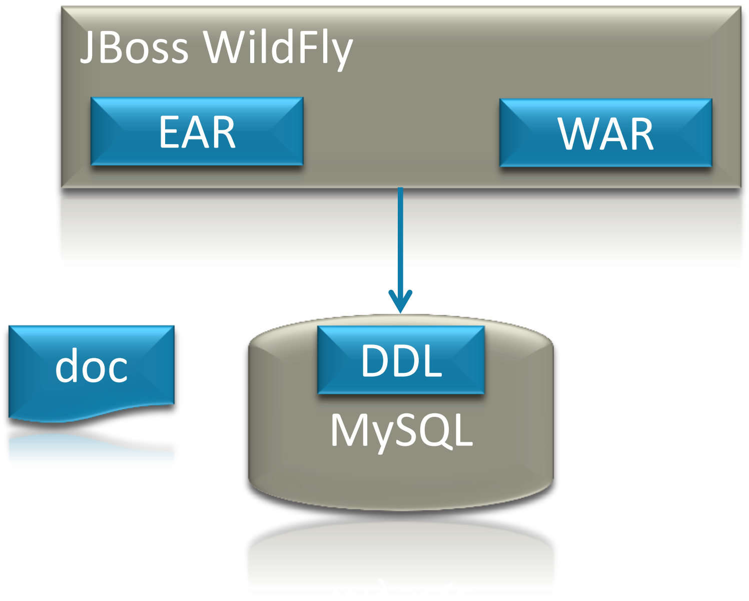
Common

- JUnit, Arquillian
- Maven
- Git / GitHub

© PRODYNA AG 2014

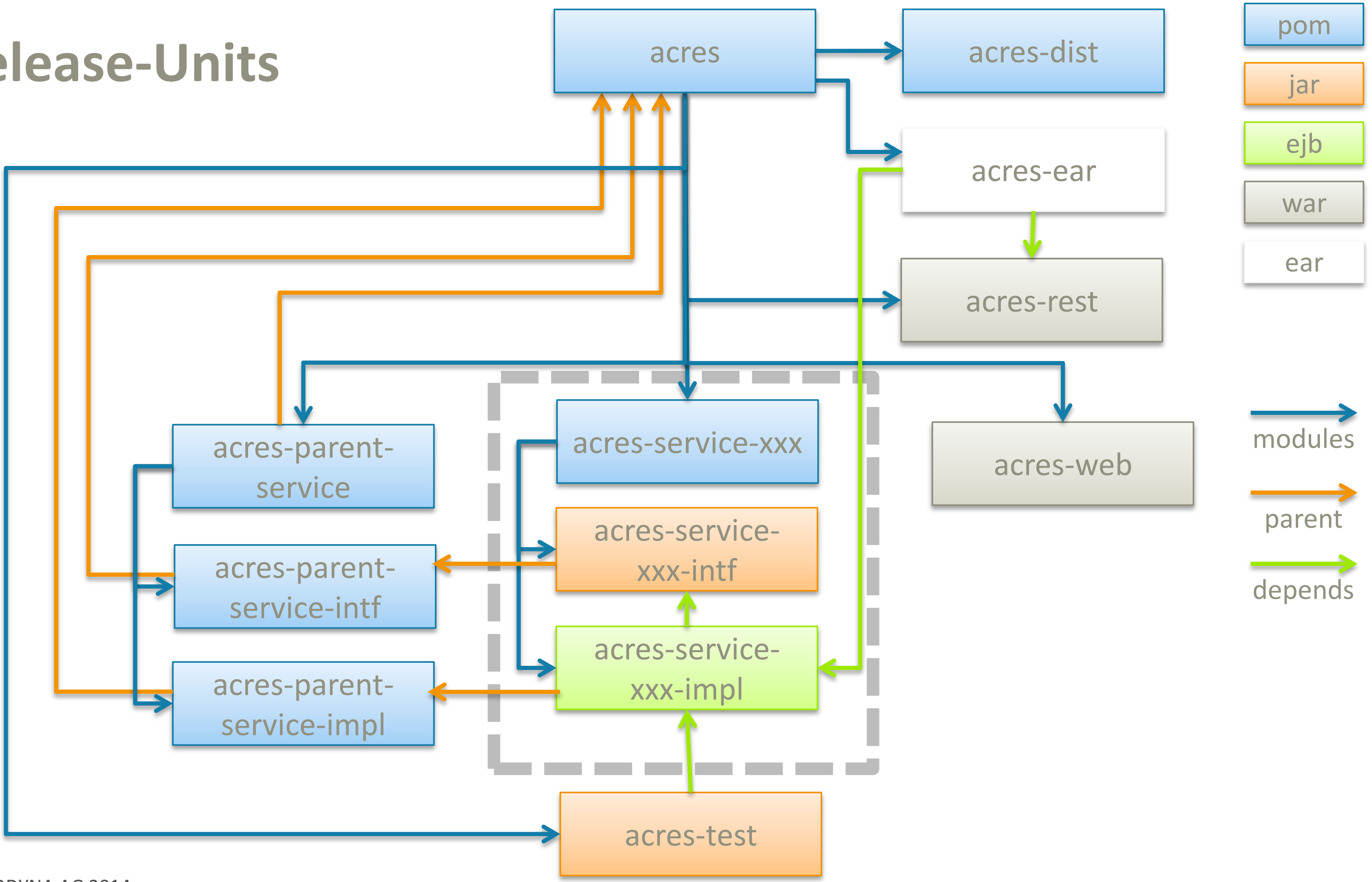
Deployment-Units

- Currently one release unit (acres-dist)
  - EAR
    - Services
    - Web (REST + HTML + JS)
    - Monitoring
  - WAR (client)
  - Database scripts
  - Environment configuration + setup instructions (README.md)



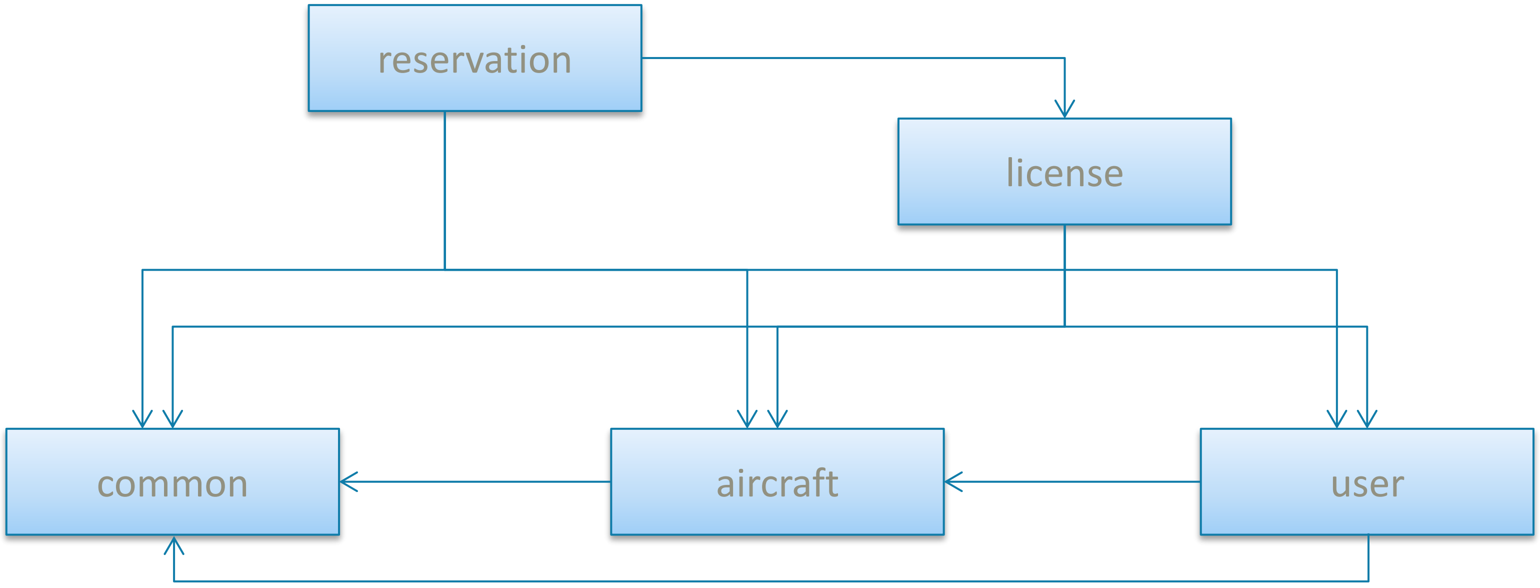
© PRODYNA AG 2014

Release-Units



© PRODYNA AG 2014

Services



© PRODYNA AG 2014

- Overview
- **Demo**
- Coding
- Non-functional requirements
- Feedback

© PRODYNA AG 2014

- Overview
- Demo
- **Coding**
- Non-functional requirements
- Feedback

© PRODYNA AG 2014

Coding: Naming Guidelines

Packages

- Base: com.prodyna.pac.acres…
- Service xxx: com.prodyna.pac.acres.xxx

Classes

- Entity/DTO: Xxx
- Service Interface: XxxService
- Service Implementation: XxxServiceBean
- REST Service Interface: XxxRestService
- REST Service Implementation: XxxRestServiceResource

Service Methods

- CRUD:
  - createXxx
  - readXxx
  - readAllXxxs
  - updateXxx
  - deleteXxx
  - searchXxx

Tables

- acres_<service>_<entity>

© PRODYNA AG 2014

PRODYNA
information architects

Coding: Entity / DTO

- Serializable
- @Enitity, @Table
- @XmlRootElement
- javax.validation Annotations (e.g. @NotNull)
- javax.xml.bind Annotations (e.g. to suppress serialization of hashed password back to client)

```
@Entity
@Table(name = "acres_user_user")
@XmlRootElement(name = "user")
public class User implements Serializable {

	private static final long serialVersionUID = 1L;

	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private Long id;

	private String name;
	@NotNull
	private String userName;
	private String email;
	private String password;

	//...
}
```

© PRODYNA AG 2014

Coding: Service Interface

- Plain Java Interface
- @Valid (also enables validation in REST service)

```
public interface UserService {

    User createUser(@Valid User user);

    User readUser(long id);

    List<User> readAllUsers();

    User updateUser(@Valid User user);

    void deleteUser(long id);
}
```

© PRODYNA AG 2014

Coding: Service Implementation

- @Stateless
- Qualifier: @Unsecured, @Monitored, @Logged
- @Inject
  - EntityManager
  - Log
  - possibly other services via their interfaces

```
@Unsecured
@Stateless
@Logged
@Monitored
public class UserServiceBean implements UserService {

        @Inject
        private Logger log;

        @Inject
        private EntityManager em;

        @Override
        public User createUser(User user) {
        // ...
                return user;
        }

        //...
}
```

© PRODYNA AG 2014

Coding: REST Service Interface

- extends Service Interface
- @Path
- @Produces, @Consumes
- Methods
  - @GET
  - @POST
  - @PUT
  - @DELETE
- @RollesAllowed, @PermitAll, @DenyAll

```
@Path("user")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public interface UserRestService extends UserService {

      @GET
      @RolesAllowed("admin")
      @Override
      List<User> readAllUsers();

      @GET
      @Path("{id:[0-9][0-9]*}")
      @RolesAllowed("admin")
      @Override
      User readUser(@PathParam("id") long id);

      @POST
      @RolesAllowed("admin")
      @Override
      User createUser(User user);
      //...
}
```

© PRODYNA AG 2014

Coding: REST Service Implementation

- @RequestScoped
- @Inject @Unsecured service
- Delegate methods

```
@RequestScoped
@Logged
public class UserRestServiceResource implements UserRestService {

        @Inject
        @Unsecured
        private UserService service;

        @Override
        public List<User> readAllUsers() {
                return service.readAllUsers();
        }

        @Override
        public User readUser(long id) {
                return service.readUser(id);
        }


        // . . .
}
```

© PRODYNA AG 2014

- Overview
- Demo
- Coding
- **Non-functional requirements**
- Feedback

© PRODYNA AG 2014

NFR: Logging / Monitoring

- Implemented in CDI interceptor (in common module)
- Attached with custom annotations
  - @Monitored
  - @Logged
- Monitoring Statistics colled by MXBean

© PRODYNA AG 2014

NFR: Security

- @Unsecured EJBs
- Web-Application Security (configured in web.xml)
  - HTTP Basic Authentication
  - Resteasy RoleBasedSecurityInterceptor
- Security Domain
  - Database login (loads user and roles)
- Client is *always* basic-authenticated
  - Replace "guest" authentication with real one on login
  - Inspired by: http://olefriis.blogspot.de/2014/01/http-basic-authentication-in-angularjs.html

© PRODYNA AG 2014

NFR: User Context

- @Inject @Current User
- @Statless CurrentUserProducerBean
  - @Inject @Unsecured UserService
  - @Resource SessionContext
  - @Produces @Current User
    - SessionContext.getCallerPrincipal()
    - UserService.findUser()
- Injected for each invocation of the @Stateless EJB: http://stackoverflow.com/a/8720148

© PRODYNA AG 2014

NFR: Exception Handling

- Possible Exceptions (all RuntimeExceptions)
  - AcresException: Business exception thrown actively by services
  - ValidationException: Thrown by bean validation framework
  - EJBTransactionRolledbackException : Thrown when a transaction cannot be committed due to a exception in a EJB method
  - Other RuntimeExceptions
- Declare RuntimeException as ApplicationException in ejb-jar.xml (to avoid wrapping all exceptions in EJBException)

- REST Exception Mapper
  - ForbiddenException: HTTP 403 (build-in)
  - ValidationException: HTTP 400 (build-in)
  - AcresException: HTTP 400
  - EJBTransactionRolledbackException
    - HTTP 400 for DB constraint violations
    - HTTP 500 otherwise

© PRODYNA AG 2014

NFR: Testing

- JUnit tests in service implementations (where necessary)
- Integration tests (acres-test) for all services based on Arquilllian and ShrinkWrap:
  - AbstractAcresTest for server-side EJB tests (without security)
    - Sets test user in LoginConfiguration to inject @Current User in services
  - AbstractAcresRestTest for REST client tests with web service security
    - Sets HTTP Basic authentication header in ClientRequestFilter

© PRODYNA AG 2014

- Overview
- Demo
- Coding
- Non-functional requirements
- **Feedback**

© PRODYNA AG 2014

PRODYNA
information architects

09.07.2014

Thank you!