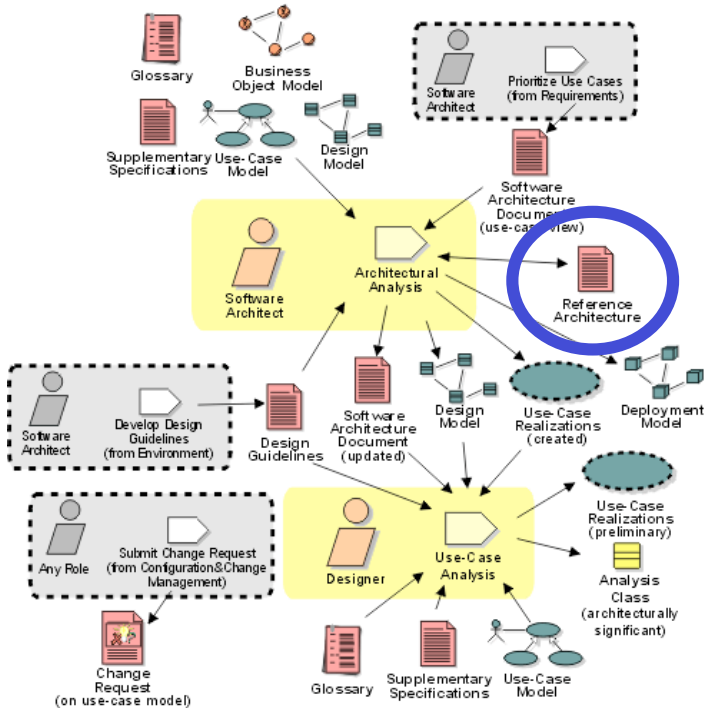
Agenda

- Why a reference architecture?
- What is it?
- How does it look?
- How is it used?
- How many do you need?
- How is it maintained?
- Where can I buy it?
- What can I expect from it?
- How does Callista do it?



CALLISTA

Why a reference architecture?

- Some areas where a RA may contribute:
  - Reduce project set-up time
  - Minimize architectural and technical re-factoring
  - Reduce cost / time of application development
  - Increase product quality
  - Secure operations characteristics of application
  - Reduce cost for maintenance

CALLISTA

Example from AstraZeneca

- Ordering system for automated chemical store
  - 5 developers - all from previous J2EE projects within AZ
  - 2 years, 4 releases
  - Technically challenging
  - Complex business rules (store optimization)
  - Exceptional transaction throughput requirements (events generated by high-speed robots 24x7)
  - Applied RUP and test-driven development
- External (Volvo IT) Reference Architecture was applied
- A review was conducted (Questionnaire) close to project closure

CALLISTA

Outcome of using RA at AZ

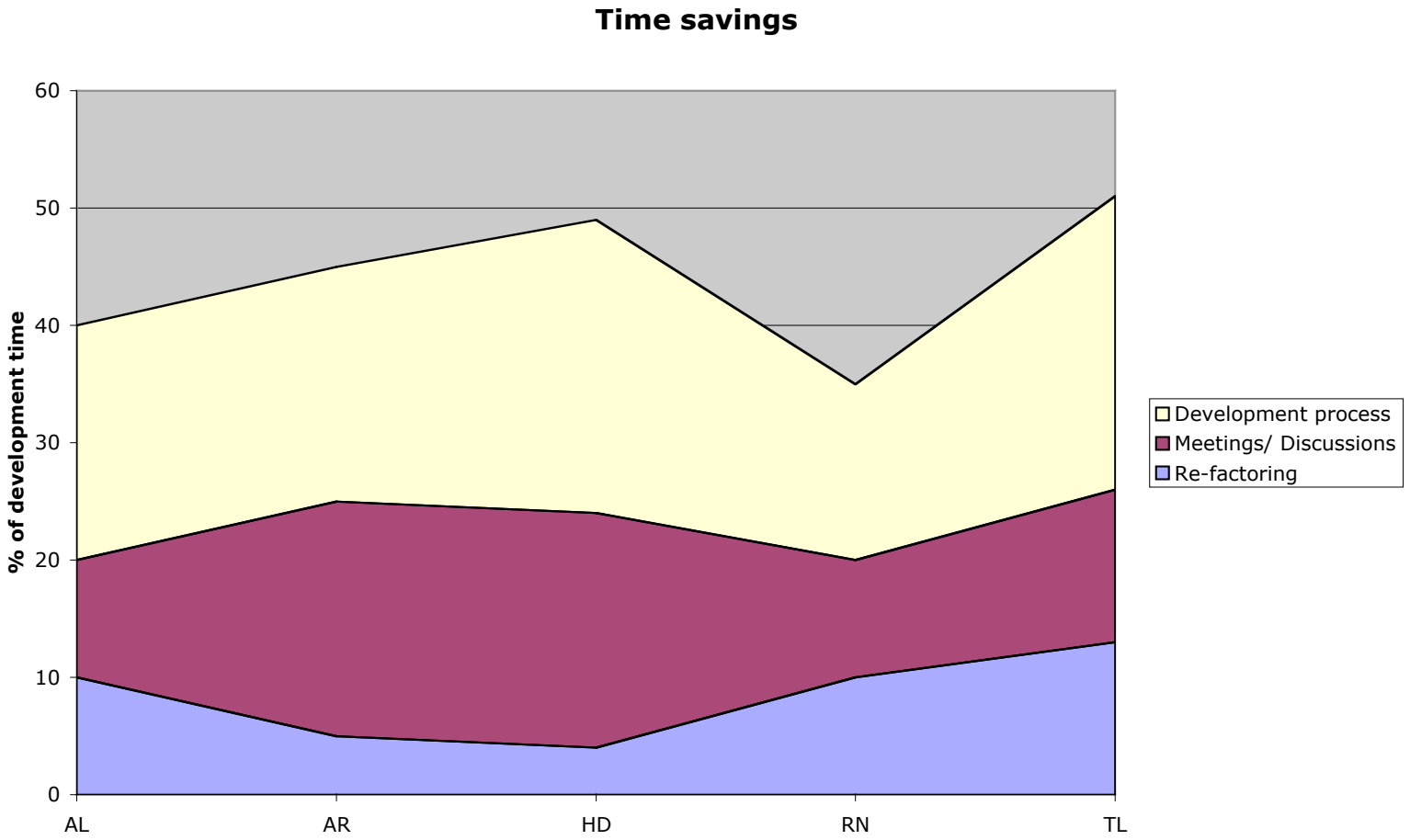
How much time did we gain, in terms of reduced architectural and technical re-factoring (%)?

How much time did we gain, by reducing the time spent discussing J2EE design issues (%)?

How much time did we gain, by cutting development time (%)?

CALLISTA

Outcome of using RA at AZ



CALLISTA

Outcome of using RA at AZ

5 times the investment in a RA was harvested within a single project.

CALLISTA

None-technical facts

- RA architect was full-time project architect and system analyst
  - In practice, it was an in-house RA
- The application is not typical for AZ, but a perfect match for the RA - only licensed for this project.

CALLISTA

What is it - Rational?

"A Reference Architecture is, in essence, a predefined architectural pattern, or set of patterns, possibly partially or completely instantiated, designed and proven for use in particular business and technical contexts, together with supporting artifacts to enable their use. Often, these artifacts are harvested from previous projects"

CALLISTA

What is it - OMG?

“The principles and the means to most effectively achieve a design vision”

CALLISTA

What is it - in practice?

- A defined set of architectural guidelines to be shared across applications and systems
- A standardized terminology (component, application, system, service etc)
- It is a vehicle to support technical decision making
- Preferably divided into External and Internal RA…

CALLISTA

What is it - External RA?

- The architectural principles that defines the structure of the system portfolio
- Rules for inter-system integration
- Defines “macro architecture” - also called “city planning”
- Should our portfolio comply to…
  - Loosely coupled Autonomous Systems (“VBS”)?.. or
  - Process-oriented applications using domain services (“SOA” - compare to IRM)?
- How is the support organization tied into the concept of systems or domains?

CALLISTA

What is it - Internal RA?

- Rules for partitioning of systems into subsystems
- Rules for partitioning subsystems into layers
- A compatible selection of design patterns to define how the architecture is to be realized in practice
- Tooling to automate the creation of code from design patterns (e.g.MDA tools)
- Standards and guidelines for one or more technical platforms
- Code framework to help the programmer adhere to guidelines and patterns.
- Standardized set of components for common mechanisms (logging, database access etc)

CALLISTA

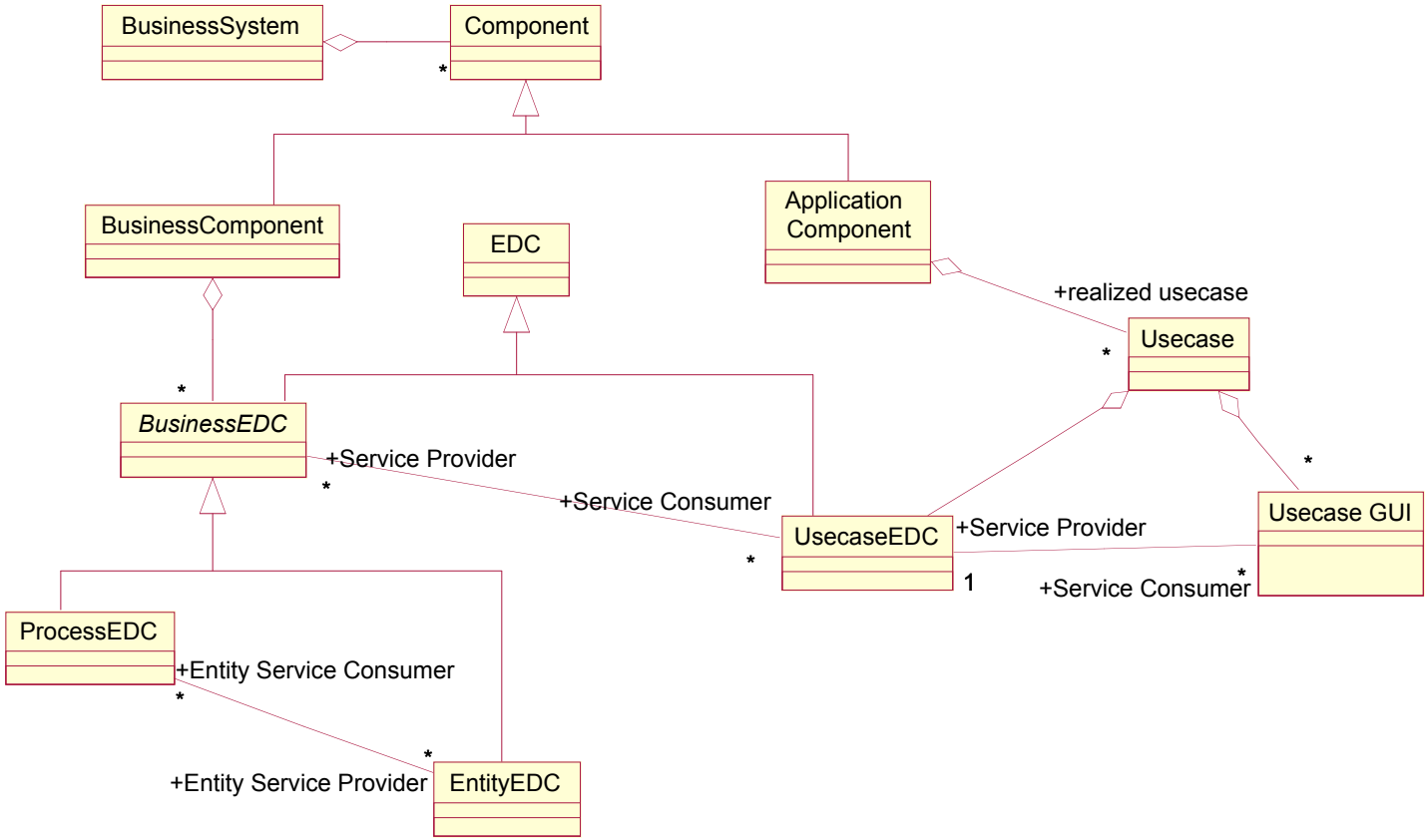
How does it look?

- Example: Jeeves RA developed at Volvo IT
- Details on the Jeeves project (formerly Japp)
  - http://www.callista.se/enterprise/resources/japp_neodeveloper_2002.pdf
- External architecture:
  - “Autonomous system”
  - Not elaborated by Jeeves project.

CALLISTA

How does it look - Subsystems?

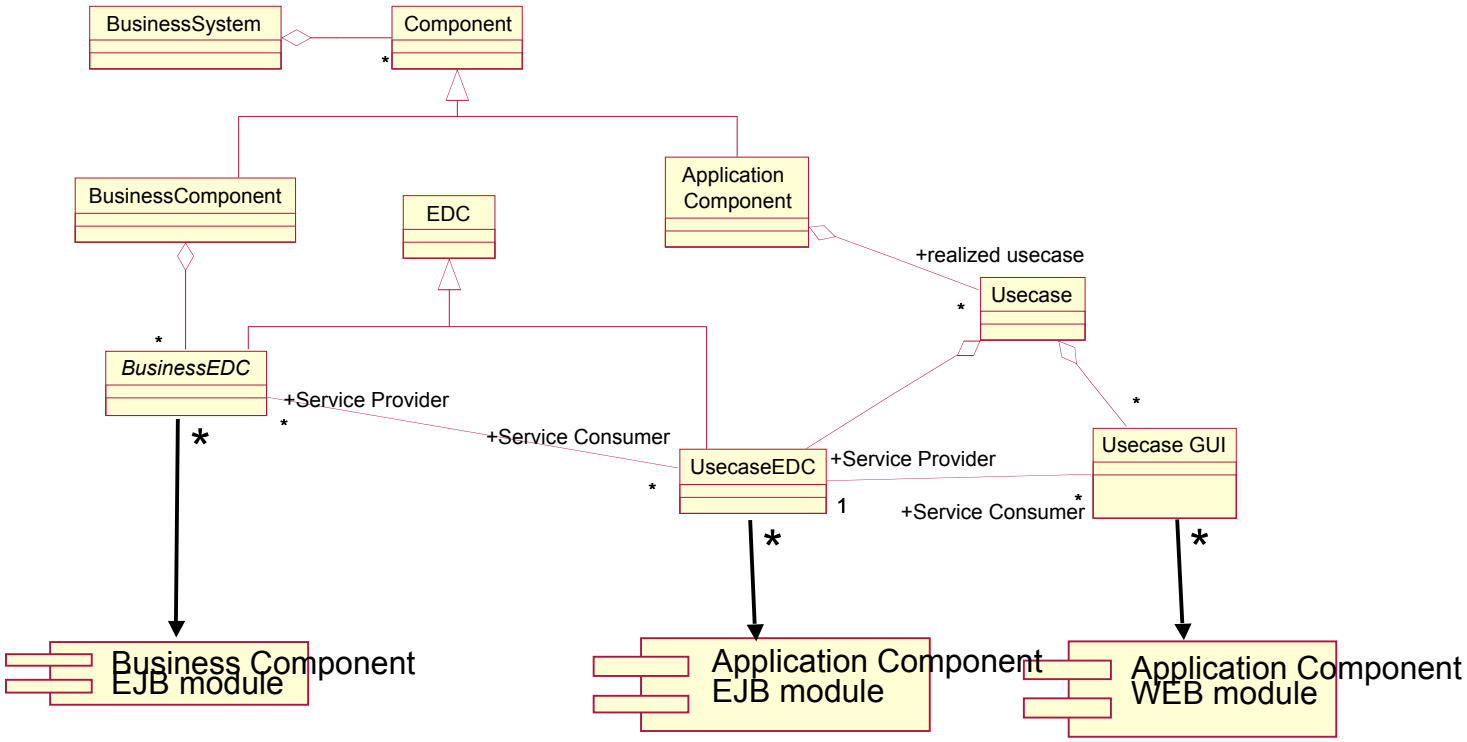
- UML model defines the rules for subsystem partitioning



CALLISTA

How does it look - J2EE modules?

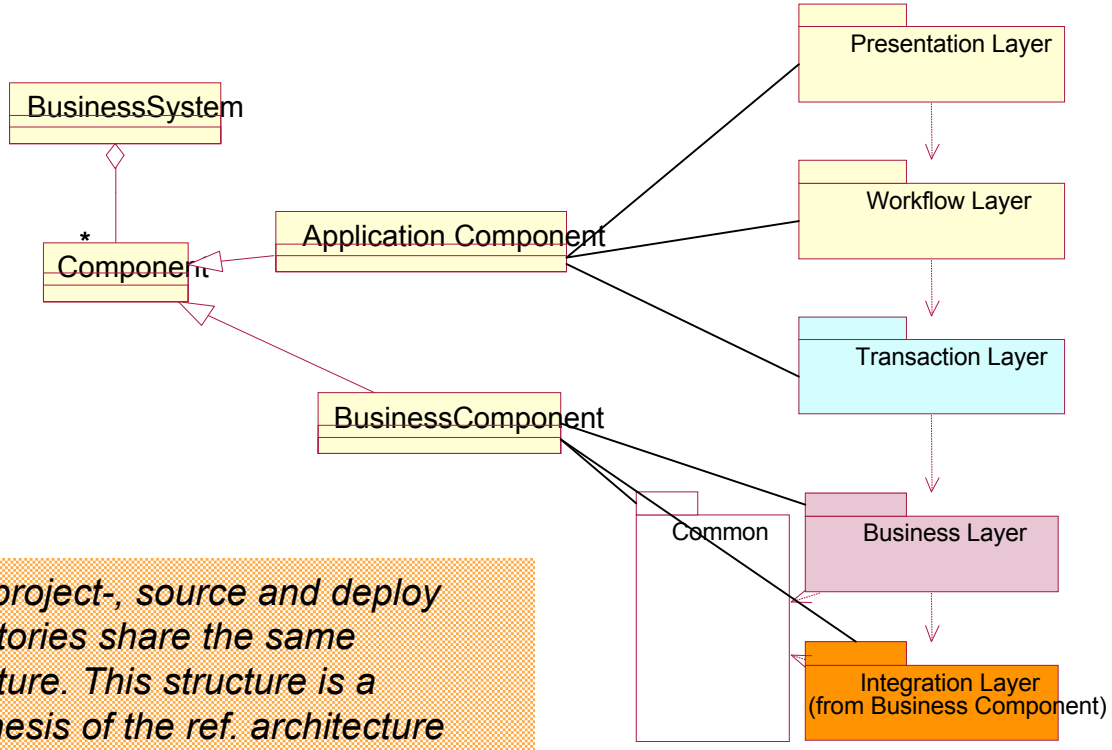
- UML model defines mapping to J2EE module types



CALLISTA

How does it look - Layers?

- UML model defines mapping from Components to layers

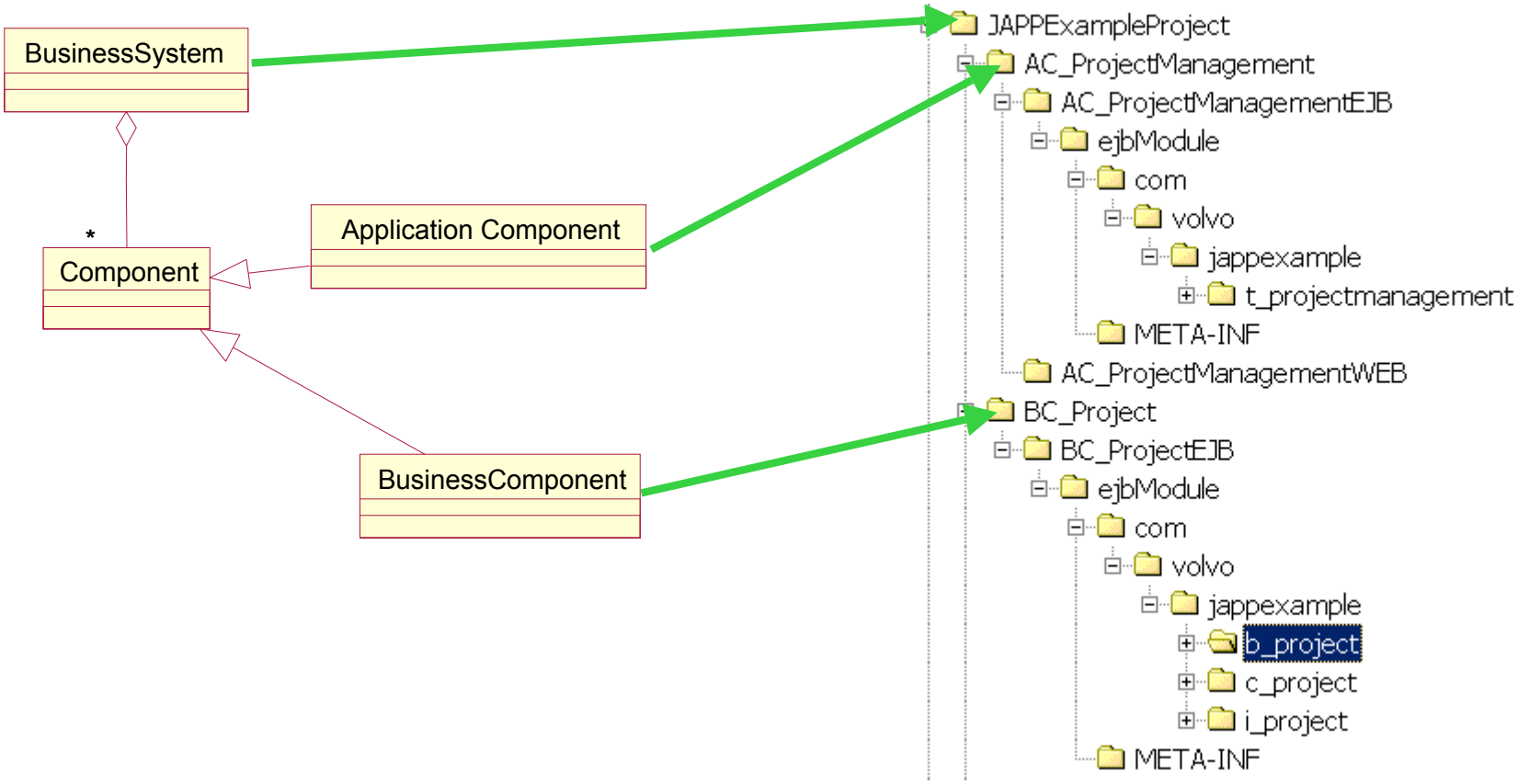


The project-, source and deploy directories share the same structure. This structure is a synthesis of the ref. architecture and the logical layers.

CALLISTA

How does it look - Naming standard?

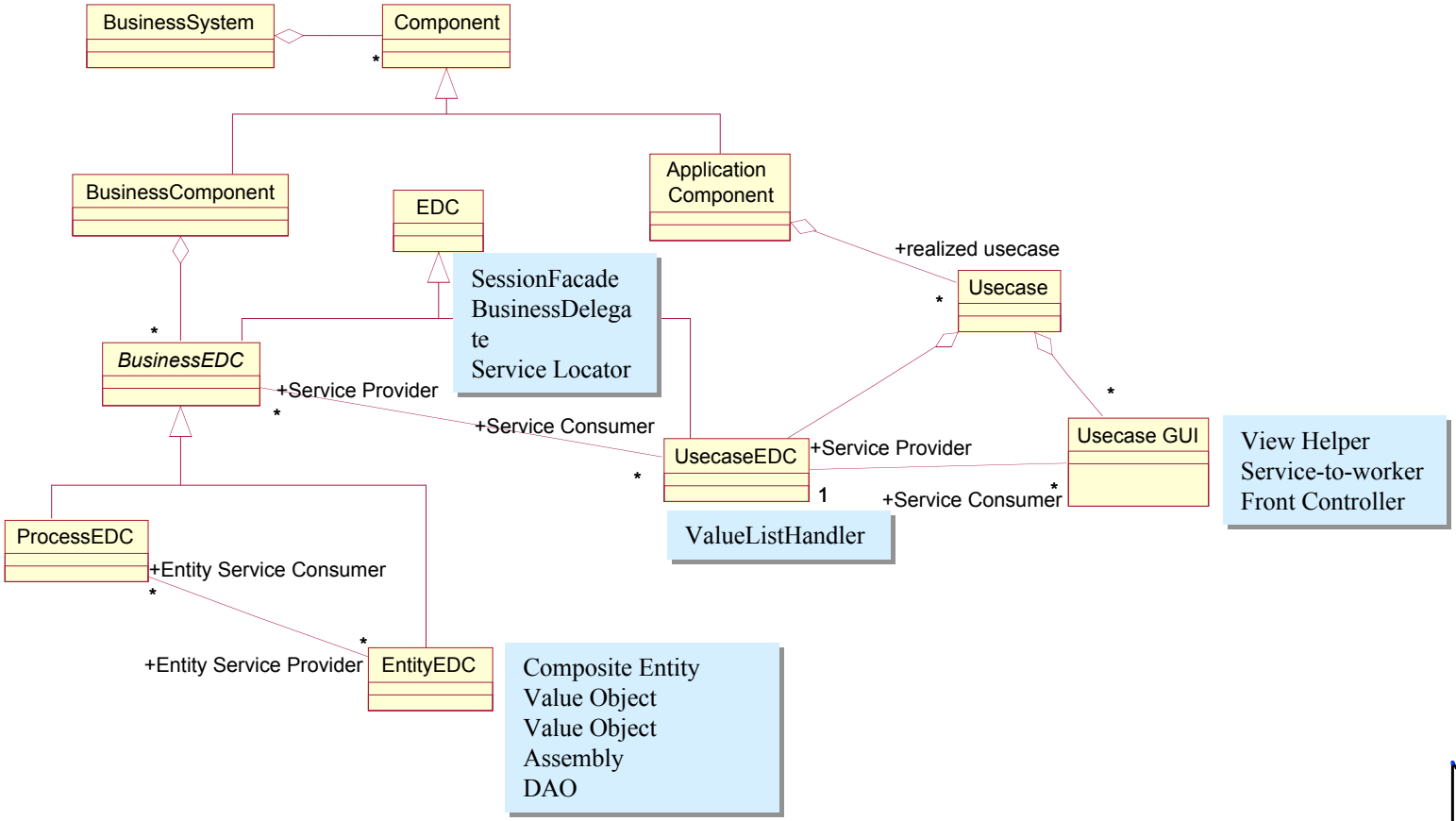
- UML model defines mapping RA to source structure



CALLISTA

How does it look - Design patterns?

- UML model defines mapping of design patterns



CALLISTA

How does it look - RA specification?

- 50-page document defines the RA

3. Business System Architecture 8
- 3.1 Structure 9
  - 3.1.1 Component 9
  - 3.1.2 Enterprise Distributed Component (EDC) 9
  - 3.1.3 Application Component 9
  - 3.1.4 Business Component 10
- 3.2 Design aspects 10
- 3.3 Relation to JEEVES 11
- 3.4 Relation to J2EE 11
- 3.5 Naming conventions 14
  - 3.5.1 Java package names 14
  - 3.5.2 Configuration files 15
- 3.6 Example 15

4. Application Component architecture 16
- 4.1 Structure 17
  - 4.1.1 Presentation Layer 17
  - 4.1.2 Workflow Layer 18
  - 4.1.3 Transaction Layer 18
- 4.2 Design aspects 18
- 4.3 Relation to JEEVES 18
- 4.4 Relation to J2EE 18

CALLISTA

How does it look - RA specification?

- A document for each pattern
  - defines implementation strategy only
  - The pattern itself is already documented by Sun

5. Composite Entity Pattern 19
   - 5.1 Motivation 19
   - 5.2 Requirements 19
     - 5.2.1 Observations 19
     - 5.2.2 Conclusions 20
   - 5.3 Solution 21
     - 5.3.1 Design model transformation 22
     - 5.3.2 Behavior of entities 25
     - 5.3.3 Example of composite entity behavior 31
     - 5.3.4 Optimistic locking 31
   - 5.4 Framework support 31

CALLISTA

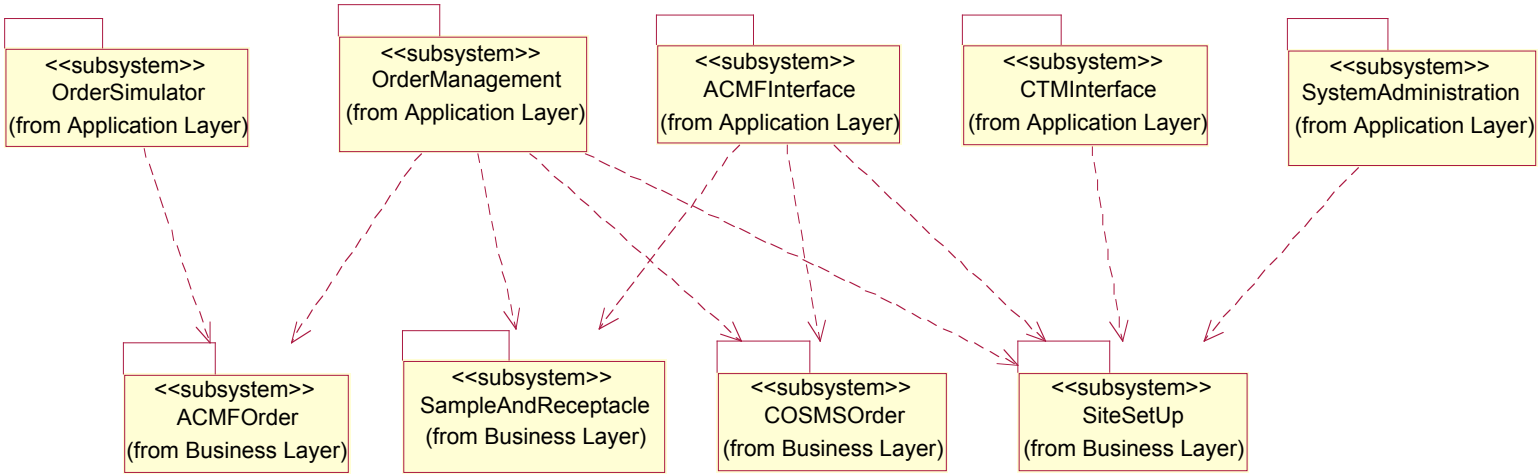
How is it used?

- How did the RA affect various roles at AZ project?
  - Software Architect
  - Systems Analyst
  - Designer
  - Programmer / Unit tester

CALLISTA

How is it used - Software Architect?

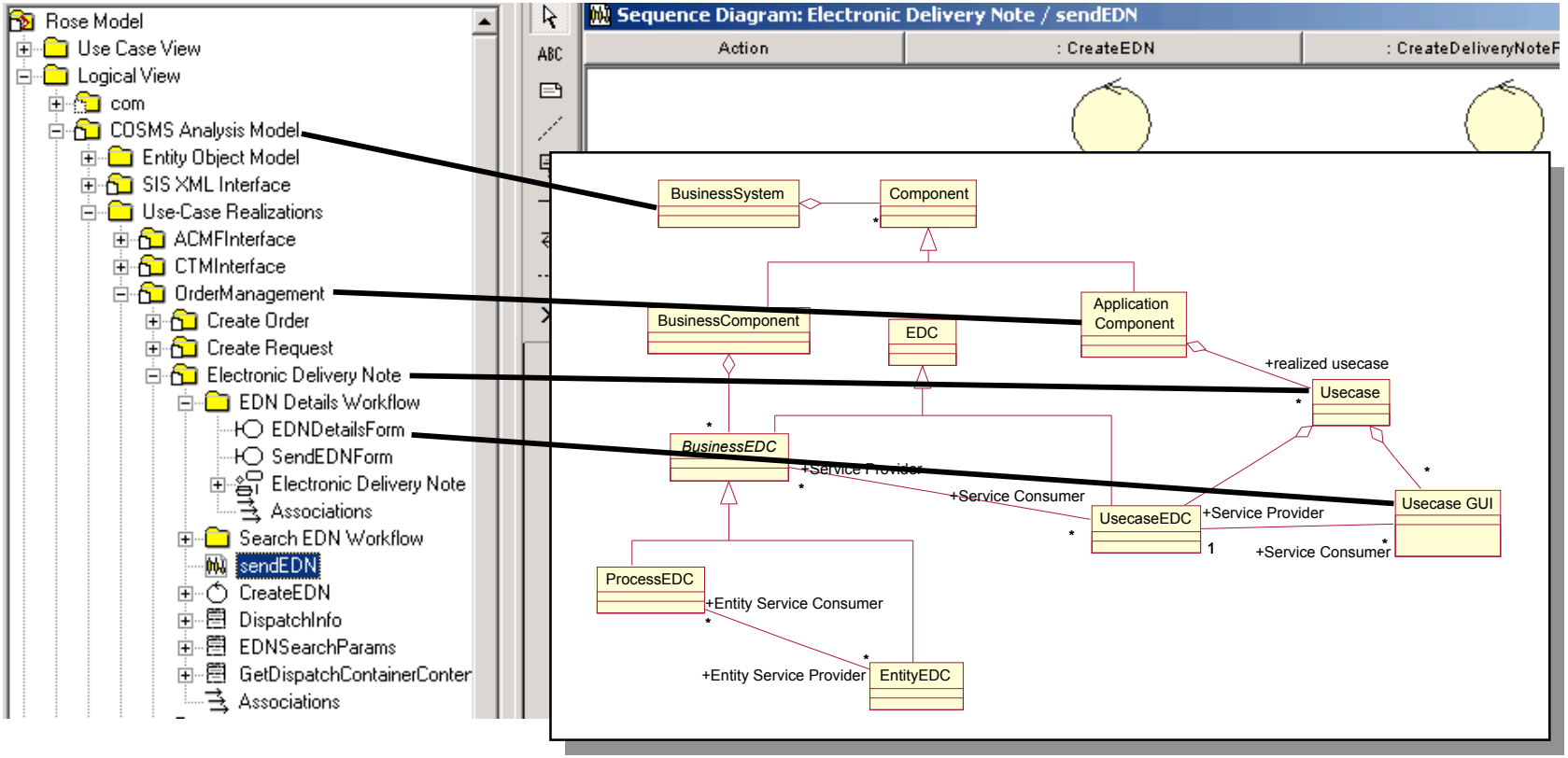
- Architect and Analyst defined Business System, Application components and Business components
- Allocates each use-case to an application component
- Many parts of SAD can be left as references to RA documentation



CALLISTA

How is it used - Systems Analyst?

- Analysis model partitioned by subsystems and use-cases, driven by RA



CALLISTA

How is it used - Designer?

- Generates code for various patterns from analysis model, according to RA
  - Composite Entity, DAO, DAO Factory, Value Object Assembly patterns
  - Transforms analysis classes into interfaces and implementation classes according to RA mapping of J2EE modules and layers

CALLISTA

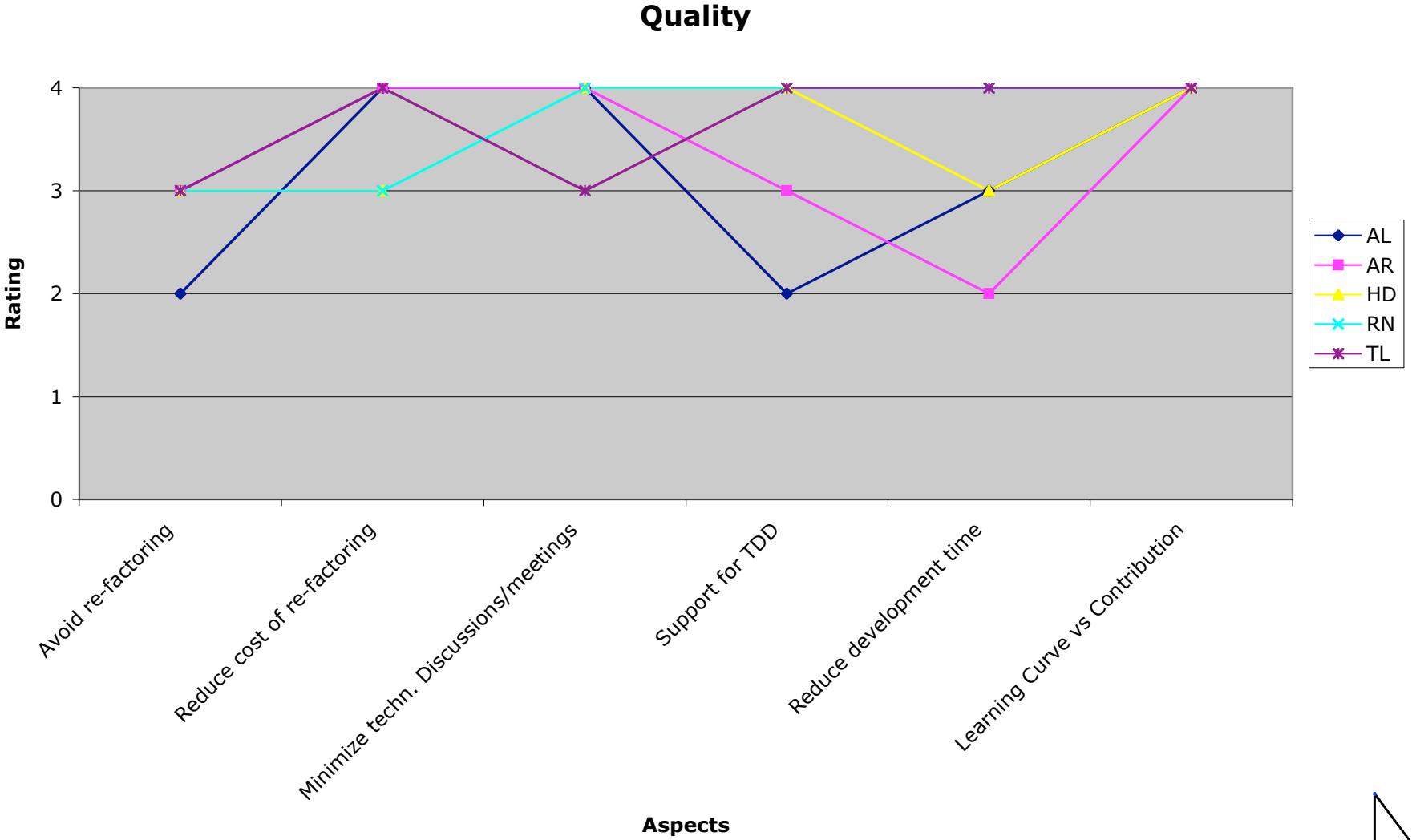
How is it used - Programmer?

- Uses RA code framework to implement classes
- RA code framework abstraction shields developer from J2EE specifics
- RA framework contains “mock” container for unit testing “out-of-app-server”

```
public void testTriggerOnCopySavedInDispenseProtocol() throws BeanAppException, InvalidNumberOfDispenseProtocolDetailsException {
    TransactionManager.getInstance().beginTransaction();
    try {
      DispenseProtocolVO vo = orderHelper.getDispenseProtocolEntity().getDispenseProtocol(new DispenseProtocolPK(301));
      impl.triggerOnCopySavedInDispenseProtocol(vo);
    }
    finally {
      TransactionManager.getInstance().rollbackTransaction();
    }
  }
```

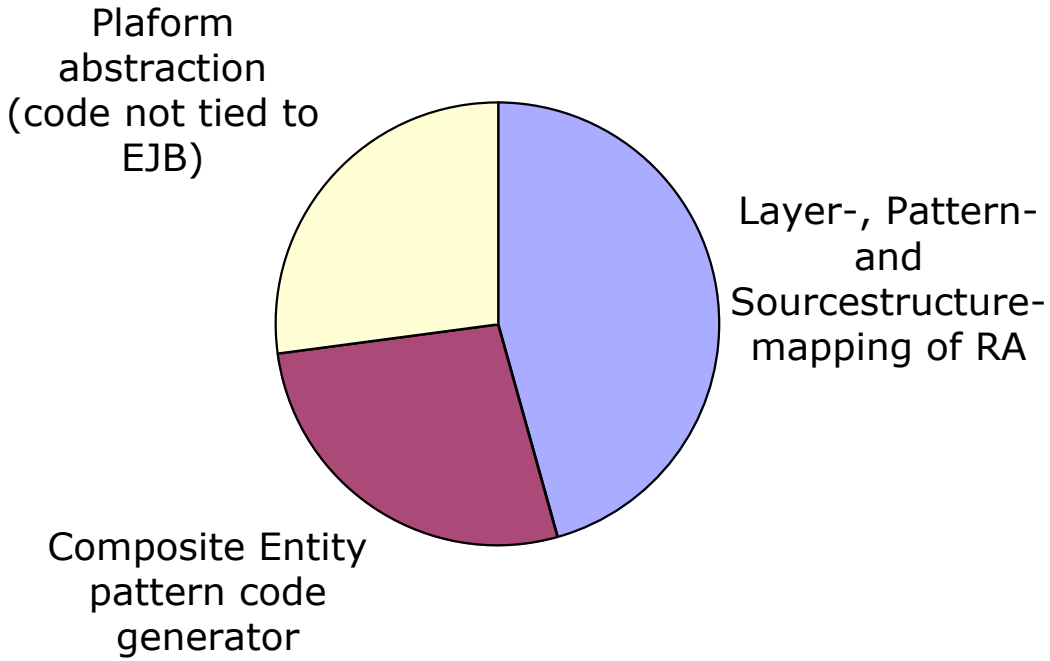

CALLISTA

How was it judged?



CALLISTA

Three most important “features”



CALLISTA

How is it used - Questionnaire

"What are the three most significant advantages achieved by using the reference architecture and supporting tools?"

- Designer:
  - applying J2EE ***design patterns*** and layering model
  - time saved with persistent entities/value objects when model changes (JE: using model-driven ***code generator***)
  - promotes a common design despite several designers
- Designer/programmer:
  - The RA rules and realizations of common ***design patterns*** minimise time and effort spent on creating them (again and again.)…
  - ***Code generation*** saves lots of time and provides reliable code when entity model is growing/being modified
  - ***Out-of-container support*** is truly a must to provide and simplify testing during development, providing a higher quality, and then greatly reduce time spent on localisation of the errors that occurs.

CALLISTA

How is it used - Questionnaire...

- Designer/programmer:
  - The support of ***running an application without a container***.
  - The RA reduces the amount of time for ***discussion*** topics, (i.e. where to place code, logging, etc).
- Designer/Programmer
  - Very explicit guidance in matters like: where do that functionality belong? what should we name that ***component/layer***? etc
  - Hides/takes care of a lot of "ugly" J2EE-matters that needs to be solved anyway, e.g. ***transactions***
  - Proven design and implementation of RA makes it possible for non-experts (i.e. people who haven't memorized the J2EE (EJB, JMS, ...) specs) to ***concentrate on solving business problems*** instead of infrastructure problems
- Programmer:
  - The ***code structure***. You know where to put your code and where to find it.
  - The ***code generator***. That all value objects and edc queries are generated is a cost/time saver.

CALLISTA

How many do I need?

- Possibly only one outer RA
- Likely a set of inner RA variants to cover different classes of applications
  - ERP type of applications (core systems of transactional nature mixing online and batch)
  - E-Business and Portals
  - Document management
  - Data analysis and reporting
  - At the detailed level, you will need to provide platform specific sections (Net, J2EE)

CALLISTA

How is it maintained?

- External RA belongs to portfolio / enterprise architect
  - Structural changes occur very rarely - “every second year”
- Internal architecture could be owned by group of project architects
  - Structural changes (UML model) happen rarely - “once a year”
  - Structural changes should be governed by a formal (slow) process.
    - This is the toughest and most critical part!
    - Don’t be buzz-word - or feature-driven - better be too slow than too fast
  - Continuous (not slow) improvements at the tooling and code framework level
  - Establish a change control process with CCB

CALLISTA

Where can I buy it?

- Should I buy it?
  - Depends on your culture
  - Taking a "complete" solution from outside is hard to anchor and to implement
- Get inspired by others work
  - SAP and other large ERP vendors have invested heavily in their RA
  - Look at others efforts as a quality assessment
- Decide what is important to you and start from there
- "It is easier to interpret the bible when you've written it your self"

CALLISTA

What can I expect from it?

- Improved quality of projects and their deliverables
- Makes life easier for maintenance organization
- Significant cut of development time
- Affective harvesting and sharing of knowledge and hard-earned experience.

CALLISTA

How does Callista do it?

- Seminars (like expansions of this one)
- Workshops to get started
- Architectural consulting to assist your RA harvesting team
- Technical management and mentoring for patterns and model-driven development
- Project architects and mentors to secure a successful career of your RA

CALLISTA