PESOS 2013

Break-Out Sessions
Process

- Workshop participants were given two post-it notes where they had to write their name and a challenge for service engineering in the cloud.
- Based on the card sorting technique, they placed each post-it note either next to a related set of post-it notes or by itself if there was no relationship to existing post-it notes.
- The resulting groups were the breakout groups.
- Each group recorded and presented the results of their discussions related to state of the practice, challenges, and potential solutions.
Group 1: Elasticity

Transcript of Post-It Notes

- Data consistency in situations of scalability and elasticity
- Leveraging standards that can enable switching from local deployments to cloud deployments (hybrid cloud)
- Dynamic resource allocation
- Properly matching service resource needs to resource allocations that will lead to economically-efficient solutions
- Dynamicity
- Scalability and feasibility
- Unlimited resources available on demand
- Reliability in light of service faults
- Anticipating runtime problems (faults, integration issues, load bursts, ...)
- Building for adaptivity: decentralized and collaborative adaptation of applications and infrastructures/platforms
- Be ready for the unexpected and adapt
- Designing for scalability
Group 2: Process
Transcript of Post-It Notes

- Testing non-functional requirements
- Coordination model (composition) and granularity
- Interoperability
- How to decompose the requirements of an application into multiple, individual and reusable services
- Service reusability
- Identifying and documenting specifics about what service is needed
- Granularity
- Understandability: clear usage guidelines and constraints
- Migration
- Extensible security models
Elasticity

Break-Out Group Report
Defining Elasticity

- Elasticity is the key differentiator of cloud computing compared to other technologies
- Elasticity is different from scalability
  - Scalability is the ability to grow to adjust to increasing loads
  - Elasticity is the ability to adjust and allocate resources according to actual demands
  - In a way, elasticity could be defined as scalability plus the ability of also scaling down
- There are two separate components that have to deal with elasticity in the cloud
  - Infrastructure component
    - Property of the infrastructure
    - Needs to monitor and react
    - Main challenge is how to do this without over-adjusting
  - Application / service component
    - Needs to be built into the applications/services
    - Main challenge is how to build applications/services that can use the elastic infrastructure
Elastic Applications/Services

• Need to be able to scale up and down

• Better if stateless
  o Otherwise, are we back to parallel computing?
  o If they are not stateless, then state needs to be able to be extracted (for instance in an external database)
    • Makes migration possible when resources need to be adjusted

• Application partitioning becomes important again
  o Separating parts of applications that need to be elastic from those that do not
  o Includes defining which parts need to be kept together so that a sub-part is not migrated and therefore it turn into a performance problem
Elasticity in PaaS vs. IaaS

• In PaaS, elasticity is inherent in the deployed applications because they are forced to follow certain design principles and accept restrictions/constraints
  o Partitioning is basically built-in in PaaS
  o Platform services used by applications are designed to be elastic

• In IaaS, elasticity has to be built into applications
  o Requires developers to build in mechanisms for creating new instances of services and killing instances when they are no longer needed
Process

Break-Out Group Report
Process Considerations

1. The cloud is about solutions
   - Solutions are driven by requirements
   - Who drives the requirements?
     - End users -> user requirements
     - Developers -> cloud requirements
   - Both types of requirements need to be considered

2. Cloud service descriptions need to be relevant for the audience (SOAML, semantic descriptions, etc.)
Process Considerations

3. Finding right granularity: trade-offs
   - Not too big, not too small
   - Considerations
     • Maintenance / Versions
     • Scalability
     • Cohesion
     • Reusability

4. Evolution / lifecycle discipline
   - Becomes more important because the amount of stakeholders increases: end users, developers, operations, cloud provider, ...
Process Considerations

5. Where to start?
   - PaaS (easy prototypes, creates lock-in)
   - IaaS (more control, much more demanding)
   - Difficult to switch!

6. Importance of Governance
   - Development process
   - Maintenance / evolution

7. Compositionality closely related to interoperability

8. Testing: Integration testing is hard and might even be infeasible
   - Agile approach from the very start
   - Keep testing in mind