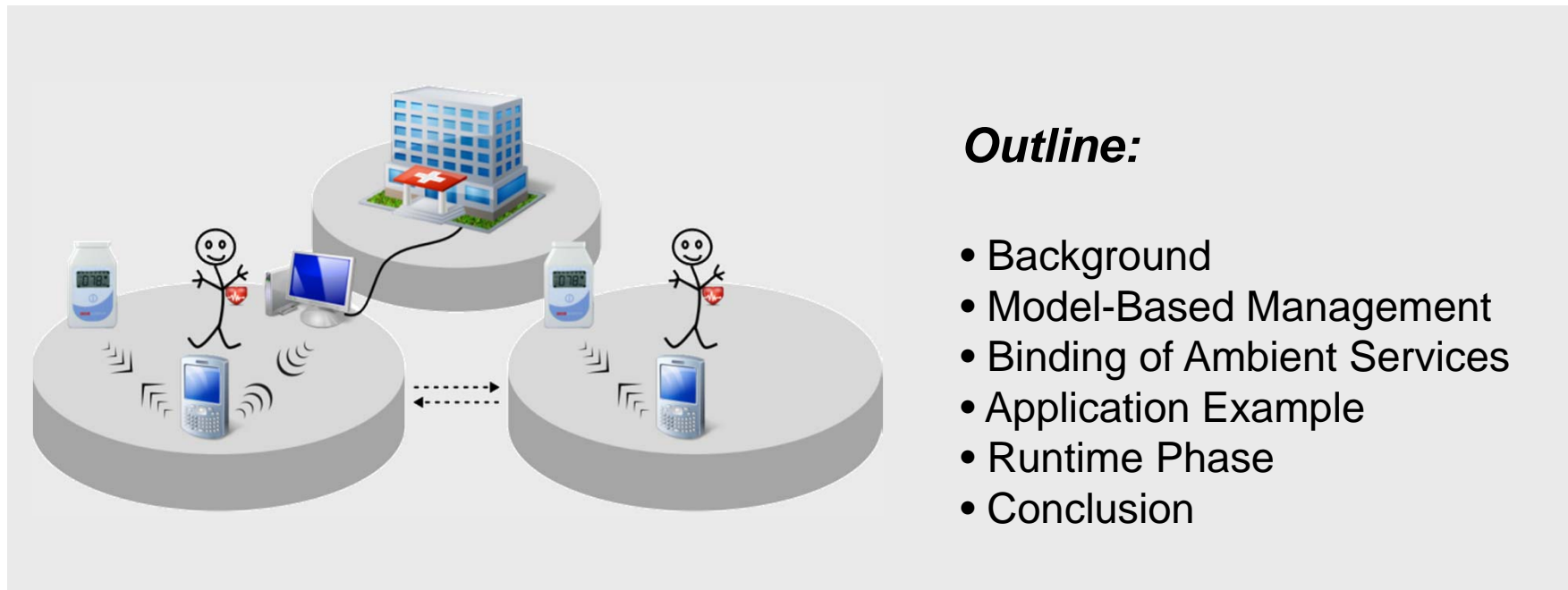


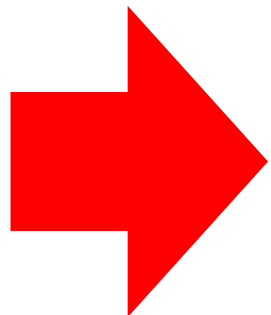
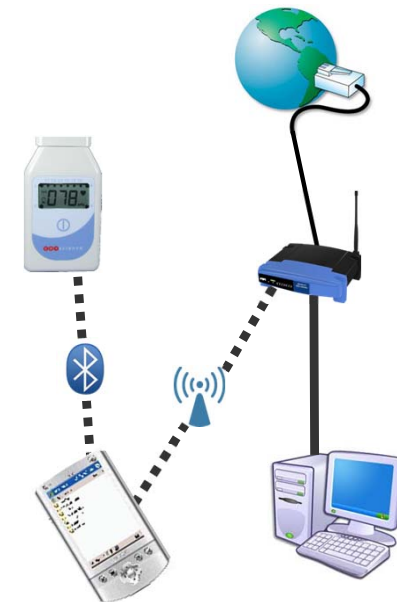
Adaptive and Reliable Binding in Ambient Service Systems



TU Dortmund University
O. Dohndorf, J. Krüger, H. Krumm

MATERNA
C. Fiehe, A. Litvina, I. Lück, F.-J. Stewing

- Ambient service systems:
 - All functionalities are provided by means of services
 - Based on distributed devices, e.g., computers, mobile phones, ...
- Challenges:
 - Potential resource-restriction of the distributed devices
 - Unpredictable exceptions due to the distribution of the system, thus
 - Spontaneously changing conditions regarding service availability, service reliability, ...

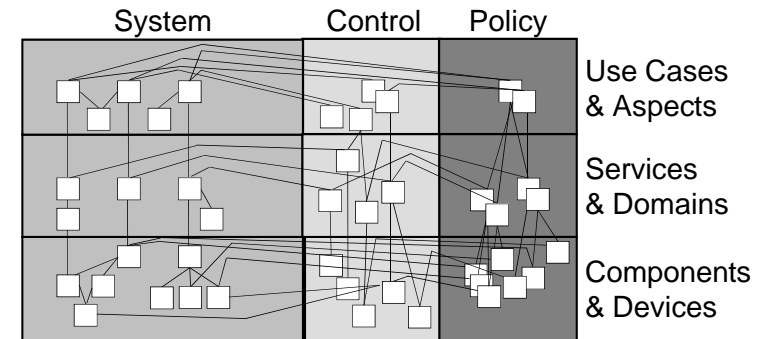


Conflict to the requirement of continuous & faultless system function, as required e.g. in the healthcare sector

- Solution: automated system management
- Goal: provide stable & reliable system behavior
 - Here: especially regarding service binding for the use of / interaction with distributed services
- Requirements:
 - Provision of rich monitoring and control functionality
 - in order to realize binding reconfiguration and system adaptation
 - Lightweightness / small footprint
 - in order to respect the potential resource restrictions of the devices used
- Approach: **Model-Based Management (MBM)**



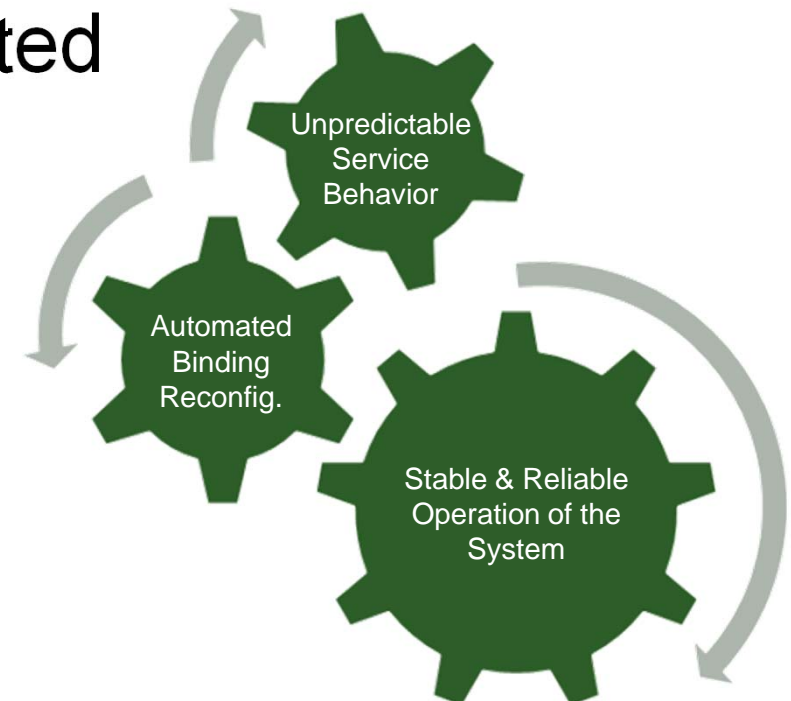
- Distinction:
 - Planning (or design) phase
 - Runtime phase
- Planning phase:
 - Based on a layered system model
 - Design of the managed system & the management system
 - Design & configuration of the adaptation functions in respect to the system structure and abstract high-level requirements
 - Automated refinement of technical low-level policies
- Runtime phase:
 - System monitoring by the management
 - Efficient enforcement of the refined policies



What are policies?
„Policies are rules governing the choices in behaviour of a system.“
M. Sloman,
Policy Driven Management for Distributed Systems,
Journal of Network and Systems Management,
Vol. 2, 1994

→ Supports a light-weighted runtime management system!

- Key challenges for service binding:
 - Unpredictable (dis-) appearance of mobile devices & thus their provided services
 - Several services of the same type but differing in their provided QoS may exist simultaneously
- Requirements for the automated binding management system:
 - Hiding of service distribution
 - Dynamic service discovery
 - Dynamic service selection
 - Binding reconfiguration
 - Policy-controlled bindings

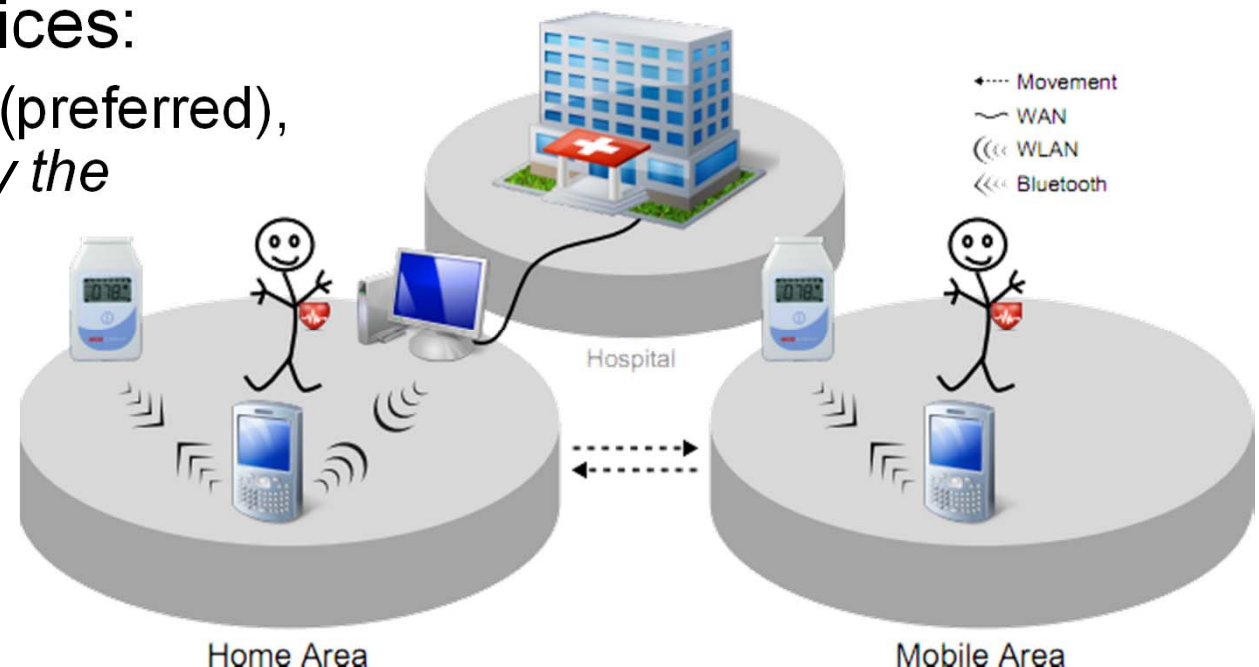


- Core: extended **Ensemble/Lease**-Concept
- Lease:
 - Usage agreement between a service and a client
 - Lease provider guarantees negotiated properties for a fixed period of time
 - Strict lifecycle:

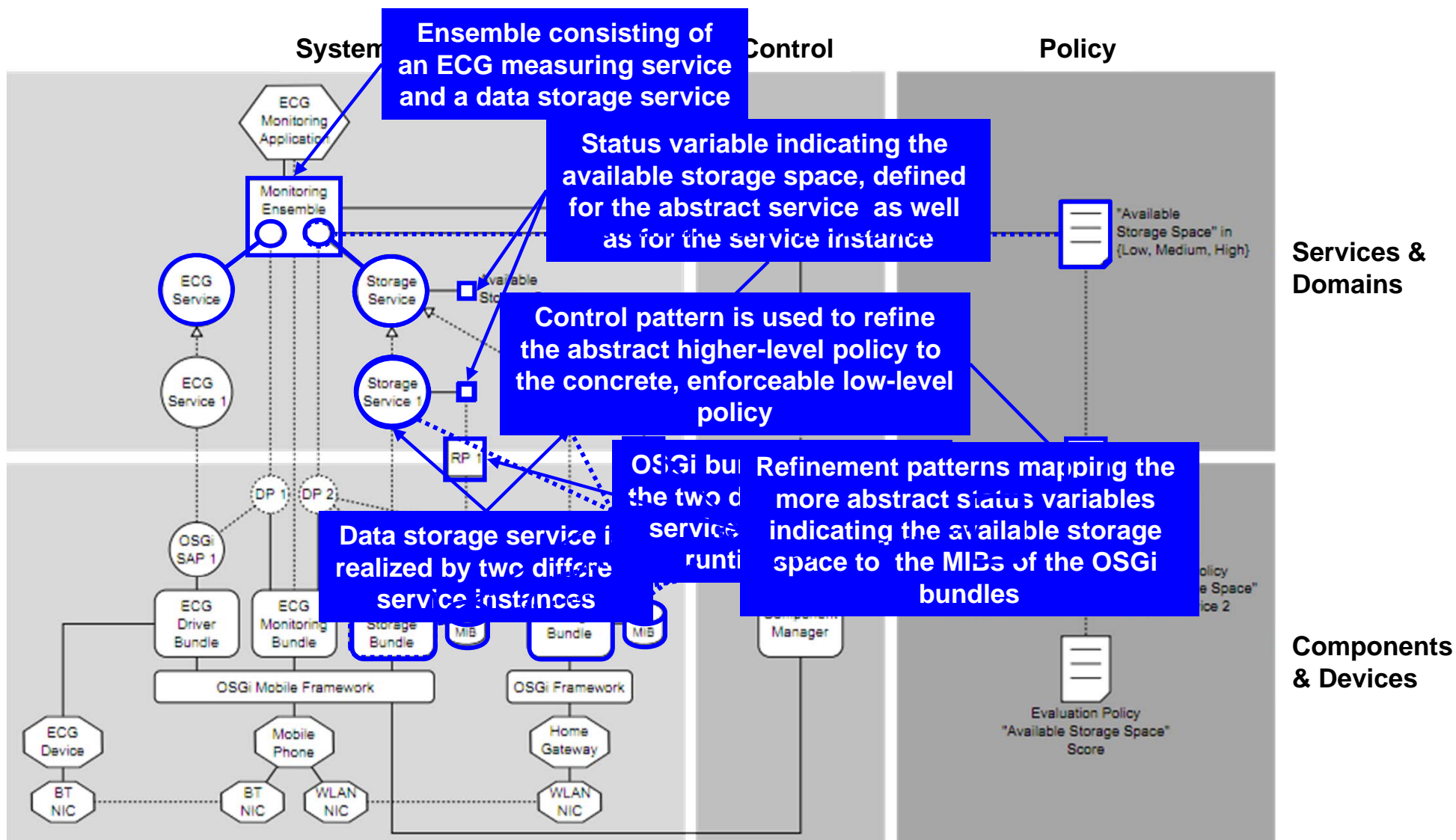


- Ensemble:
 - Logical group of services (i.e., leases) required by a client
 - Allocated / leased either as a whole, or not at all
 - Only a client can release an ensemble during the lease time (apart from exceptions)

- Scenario:
 - Cardiac rehabilitation, comparable to the OSAmI project
 - Constant recording of ECG data
 - Software execution platform & data capture device:
 - Mobile phone
 - Data storage devices:
 - “Home gateway” (preferred),
or alternatively the
 - Mobile phone
 - Both devices provide similar data storage services



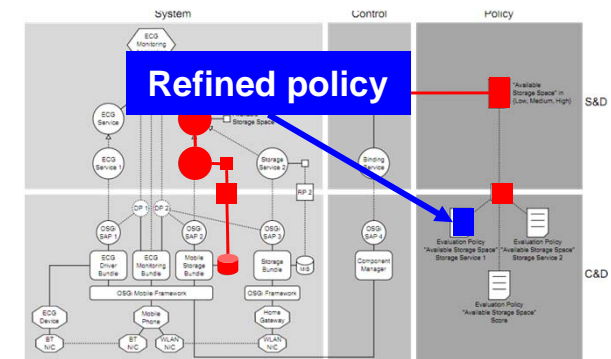
- Resulting Model (after policy derivation):



- Refined policy to analyze the available storage of the mobile storage bundle:

```
public class AvailableStorageSpace_Value_Storage_Service_1 implements PolicyEvaluation {  
    // some code omitted due to limited space  
    public Object evaluate() {  
        ManagementData data = session.getNodeValue("/BUNDLE/MOBILE_STORAGE_BUNDLE/STATUS/FREE_CAPACITY");  
  
        Integer freeCapacity = (Integer) data.getValue();  
  
        if (freeCapacity < 10) return "CRITICAL";  
        else if (freeCapacity < 30) return "LOW";  
        else if (freeCapacity < 60) return "MEDIUM";  
        return "HIGH";  
    }  
}
```

Evaluation of the numeric value of the available storage space; concrete value ranges are retrieved from the marked pattern path from the higher level policy to the low level components realizing the service



- Actual service-selection policy (not shown in the model):

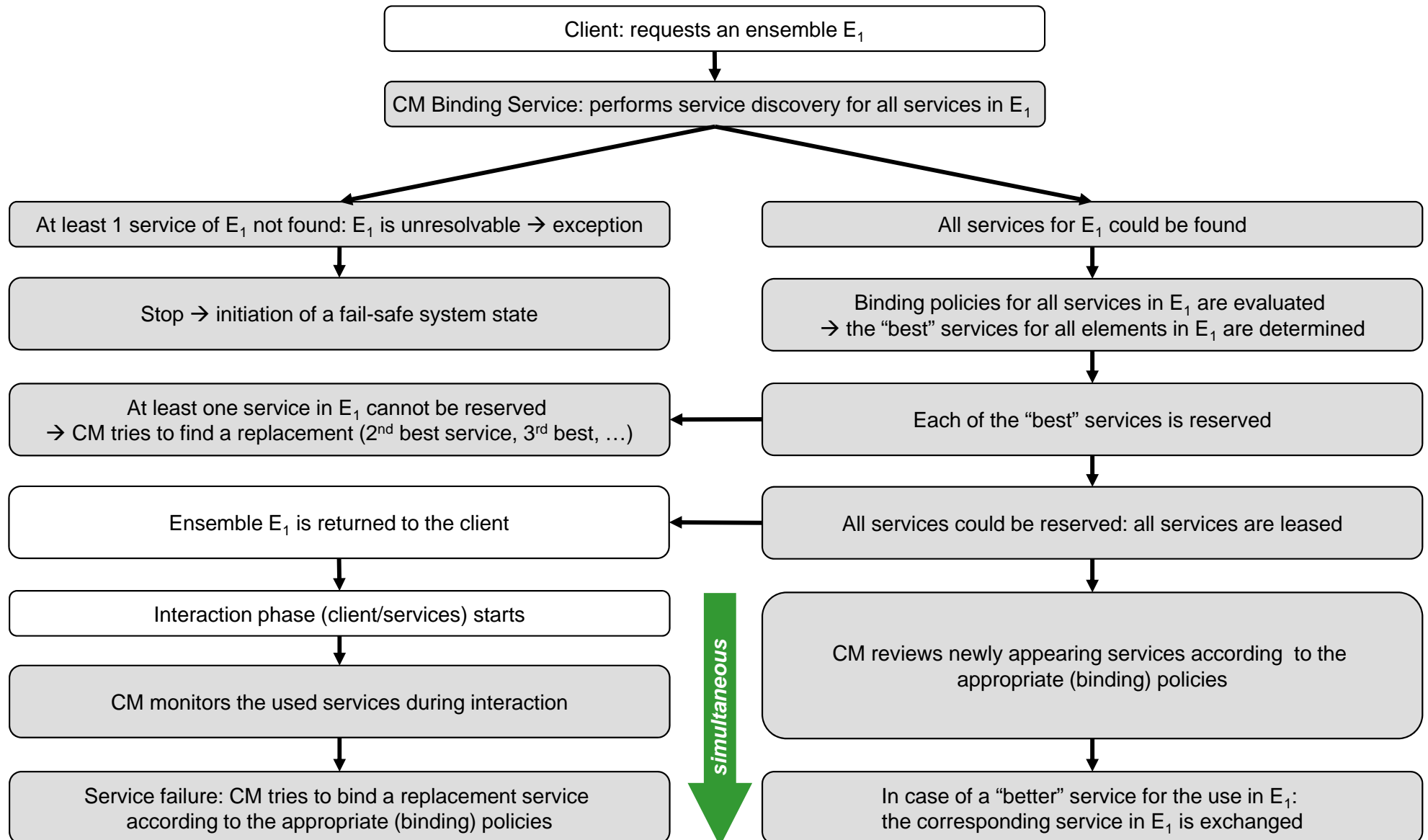
```
public class Selection_Storage_Service implements PolicyEvaluation {
    // some code omitted due to limited space
    public Object evaluate(Object... params) {
        for (Object param : params) {
            String serviceId = (String) param;
            String value = (String) policyService.requestPolicyEvaluation("AvailableStorageSpace_Value_"
                + serviceId);
            Double score = (Double) policyService.requestPolicyEvaluation("AvailableStorageSpace_Score_"
                + serviceId, value);
            Double serviceScore = alpha[i++] * score;
            if (serviceScore > maxServiceScore) {
                maxServiceScore = serviceScore;
                selectedServiceId = serviceId;
            }
        }
        return selectedServiceId;
    }
}
```

1. Traverse all available storage service instances
2. Evaluate the storage space left for the particular service (by another policy call)
3. Calculate the score value to compare the storage spaces left (by another policy call)
4. Remember the “best” currently available service
5. Return the ID of the best service to the management

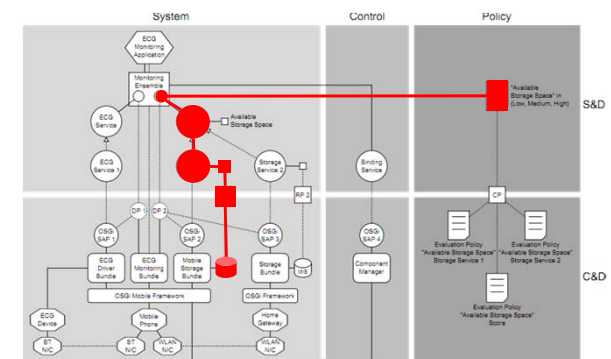
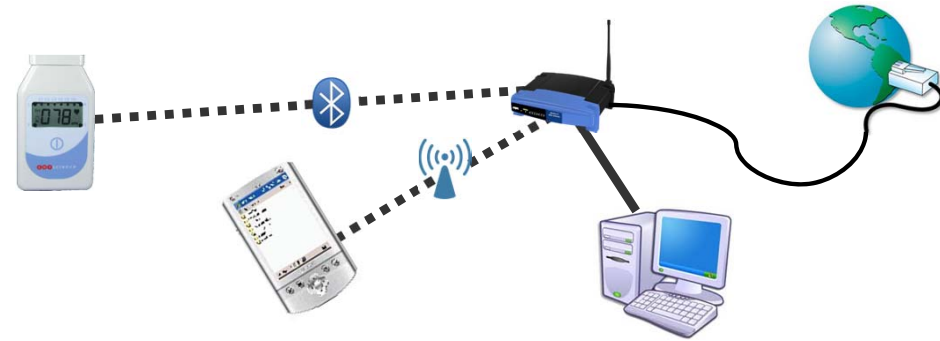
Design Phase

End of design phase: refined policies are deployed to the appropriate Component Manager (CM)

Runtime Phase



- Ambient service system: based on potentially small devices
- Challenges:
 - resource-restriction
 - exceptions
 - unpredictable behavior
- Solution: automated system management
 - Based on policies
- Approach: model-based management
 - Planning phase + Runtime phase
 - Lightweightness / small footprint
- Ensemble/Lease concept
- Application example
 - taken from the OS@ml project



Thanks for the attention!



Questions?